Error Decoding Strategies for Algebraic Codes

Gitit Ruckenstein
Error Decoding Strategies for Algebraic Codes

Research Thesis

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

Gitit Ruckenstein

Submitted to the Senate of
the Technion — Israel Institute of Technology

Kislev 5762 Haifa December 2001
The research thesis was done under the supervision of Assoc. Prof. Ron Roth in the department of Computer Science

Ronny, thank you for the professional and patient guidance, for the interesting and helpful talks we had, and for everything I learned from you as a researcher and as a teacher.

I would like to thank Abraham Lempel for the guidance in the beginning of my Ph.D. studies and for many knowledgeable advice along the way.

I thank Abraham Lempel, Ronny Roth, and Gadiel Seroussi for organizing my visits to Hewlett-Packard Laboratories, Palo Alto, CA. Those visits had a significant contribution to my research.

And I would like to thank all the other people from whom I learned and with whom I worked in the Computer Science department and in the Technion.

Warm thanks to my husband, Yaron, to my parents, Edna and Shlomo Sadeh, to my sisters, Moran and Kinneret, and to all my family members and friends for their support throughout my studies.

The generous financial help of the Technion, VATAT, the Wolf fund, and the Gutwirth fund is gratefully acknowledged.
Contents

Abstract 1

Notations and Abbreviations 3

1 Introduction 5
   1.1 Error correcting codes and their decoding 5
   1.1.1 Using block codes through discrete memoryless channels 5
   1.1.2 Maximum-likelihood decoding in the q-ary symmetric channel 6
   1.1.3 Decoding strategies for algebraic codes 7
   1.2 MDS codes and Reed-Solomon codes 8
   1.3 Decoding Reed-Solomon codes 9
      1.3.1 Classical decoding 9
      1.3.2 List decoding: Sudan’s and Guruswami-Sudan’s algorithms 11
   1.4 Efficient procedures for list decoding of Reed-Solomon codes 12
      1.4.1 Results of this work 12
      1.4.2 Related works 15
   1.5 Bounds on the list-decoding radius 16
      1.5.1 Previous work 17
      1.5.2 Results of this work 18
   1.6 Organization of this work 19

2 Technical background 21
   2.1 Sudan’s algorithm 21
   2.2 Guruswami-Sudan’s algorithm 22
   2.3 Guruswami-Sudan’s decoding radius $\left\lceil \frac{t_i(n,d)}{2} \right\rceil - 1$ 23
   2.4 The FIA 24

3 Efficient interpolation of bivariate polynomials 29
   3.1 Introduction 29
   3.2 Extended key equation 29
   3.3 From EKE to homogenous linear equations 32
   3.4 FIA-based algorithm for solving the EKE 34
   3.5 Improved algorithm for solving the EKE 37
   3.6 Computing $Q^{(0)}(x)$ 42
4 Finding linear factors of bivariate polynomials

4.1 Introduction ......................................................................................... 43
4.2 The algorithm ............................................................................... 43
4.3 Complexity analysis .................................................................... 45

5 Efficient list-decoding algorithm—summary and outline for extension

5.1 Summarizing efficient implementation of Sudan’s algorithm ............ 49
5.2 Outline for extension .................................................................. 51
5.3 Preprocessing Sudan’s equations—a second viewpoint ................. 54

6 Lower bounds on the list decoding radius of general block codes

6.1 Generalization of the GS bound .................................................. 57
6.2 $(\ell, r)$-configurations ................................................................. 58
6.3 Necessary conditions on failing lists .............................................. 61
6.4 Block designs and failing lists ....................................................... 63

7 Bounds on the list decoding radius of Reed-Solomon codes

7.1 Main Results ............................................................................... 67
7.2 Realizing failing lists in RS codes ................................................ 69
  7.2.1 The difference condition and simple sets of polynomials ......... 70
  7.2.2 Rates above $2/(\ell(\ell+1))$ ....................................................... 71
  7.2.3 The low-rate range: $0 < (k-1)/n \leq 2/(\ell(\ell+1))$ .......... 73
7.3 Proof of main results for RS codes ............................................... 75
  7.3.1 The function $\Delta_t^{RS}(n, d; q)$ .............................................. 75
  7.3.2 Types of simple sets of polynomials ..................................... 75
  7.3.3 The high-rate range: Proposition 7.1 .................................. 76
  7.3.4 The mid-rate range: Propositions 7.3–7.6 ......................... 76
  7.3.5 The low-rate range: Proposition 7.2 ................................. 81

8 Average number of codewords in list decoding of MDS codes

8.1 Introduction ............................................................................... 83
8.2 Previous work ........................................................................... 83
8.3 Average number of incorrect codewords in a list ......................... 85

9 Conclusion

9.1 Contribution of this work ............................................................ 89
  9.1.1 Efficient procedures for RS list decoding ......................... 89
  9.1.2 The decoding radius as a function of the maximal list size .... 90
9.2 Problems for further research .................................................... 91

Bibliography .................................................................................... 93
List of Figures

1.1 Error locator polynomial as a linear shift register. ......................... 11
2.1 Fundamental Iterative Algorithm (FIA). .................................. 25
2.2 Typical values taken by \((\rho, \sigma)\) throughout the course of the FIA. .... 26
2.3 Typical values of \((\rho, \sigma)\) in the FIA when the input has a Hankel form. .... 27
3.1 Multi-level shift register representing (3.18). .............................. 34
3.2 FIA-based algorithm for solving (3.18). .................................. 36
3.3 Typical values of \((\rho, (\mu, \nu))\) for \(\ell = 2\) in algorithm of Figure 3.2. .... 37
3.4 Improved algorithm for solving (3.18). .................................. 39
4.1 Recursive procedure for finding a superset of the consistent polynomials. .... 44
5.1 Summary of efficient list decoding algorithm. ............................. 50
6.1 (3, 2)-configuration over GF(11). ........................................... 60
6.2 (4, 3)-configuration over GF(16). ........................................... 61
Abstract

Error correcting codes are used to detect and correct errors that occur in data when transmitted through noisy channels. The channels considered are usually communication channels or storage devices. An \((n, M, d)\) block code over an alphabet \(F\) is an \(M\)-subset of \(F^n\) with minimum Hamming distance \(d\) between any two different elements (codewords).

A list decoder with a decoding radius \(\tau\) for an \((n, M, d)\) block code \(C\) over \(F\) maps words in \(F^n\) into sets (lists) of codewords in \(C\), such that the output list corresponding to an input word \(y\) consists of all the codewords in \(C\) at Hamming distance \(\tau\) or less from \(y\). The maximal list size is \(1\) if and only if the decoding radius is not greater than \(\lfloor (d-1)/2 \rfloor\).

An \([n,k,d]\) linear code over the finite field \(F = GF(q)\) is an \((n,q^k,d)\) code which forms a linear space over \(F\). Such a code is called \([n,k]\) MDS if \(d = n-k+1\). An \([n,k]\) Reed-Solomon (RS) code over \(F\) is an MDS code in which the codewords are obtained by evaluating the various polynomials of degree less than \(k\) over \(F\) at \(n\) prescribed different elements of \(F\). The well-known RS decoding algorithm by Berlekamp and Massey and the Euclid-based algorithm by Sugiyama et al. correct up to \(\lfloor (d-1)/2 \rfloor = \lfloor (n-k)/2 \rfloor\) errors in time complexity \(O((n-k)^2)\).

A polynomial-time list decoding algorithm for \([n,k]\) RS codes has been presented by Sudan in 1996. For codes of rate \(k/n\) not greater than \(1/3\), his algorithm corrects more than \(\lfloor (n-k)/2 \rfloor\) errors. In 1998, Guruswami and Sudan presented an improved algorithm which enables the polynomial-time correction of more than \(\lfloor (n-k)/2 \rfloor\) errors for every \([n,k]\) RS code at any code rate.

In this work, a list decoding algorithm is presented for \([n,k]\) Reed-Solomon (RS) codes over \(GF(q)\), which is capable of correcting more than \(\lfloor (n-k)/2 \rfloor\) errors. Based on the previous work of Sudan, an extended key equation (EKE) is derived for RS codes, which reduces to the classical key equation when the number of errors is limited to \(\lfloor (n-k)/2 \rfloor\). Generalizing Massey’s algorithm which finds the shortest recurrence that generates a given sequence, an algorithm is obtained for solving the EKE in time complexity \(O(\ell \cdot (n-k)^2)\), where \(\ell\) is a design parameter, typically a small constant, which is an upper bound on the size of the list of decoded codewords (the case \(\ell = 1\) corresponds to classical decoding of up to \(\lfloor (n-k)/2 \rfloor\) errors where the decoding ends with at most one codeword). This improves on the time complexity \(O(n^3)\) needed for solving the equations of Sudan’s algorithm by a naive Gaussian elimination. The polynomials found by solving the EKE are then used for reconstructing the codewords in time complexity \(O(\ell \log^2 \ell k (n + \ell \log q))\) using root-finders of degree-\(\ell\) univariate polynomials. An extension of the algorithm into an efficient implementation of Guruswami-Sudan’s algorithm is outlined as well.

Techniques are presented for computing upper and lower bounds on the number of errors
that can be corrected by list decoders for general block codes and, specifically, for Reed-Solomon (RS) codes. The list decoder of Guruswami and Sudan implies such a lower bound (referred to here as the GS bound) for RS codes. It is shown that this lower bound, given by means of the code length, the minimum Hamming distance, and the maximal allowed list size, applies in fact to all block codes. Ranges of code parameters are identified where the GS bound is tight for worst-case RS codes, in which case the list decoder of Guruswami and Sudan provably corrects the largest possible number of errors.

On the other hand, ranges of parameters are provided for which the GS lower bound can be strictly improved. In some cases, the improvement applies to all block codes with a given minimum Hamming distance, while in others it applies only to RS codes.

Upper bounds are derived on the average number of incorrect codewords (i.e., codewords different from the transmitted one) that are included in the output list of a decoder with radius $\tau$ for any $[n,k]$ MDS code, where $[(n-k)/2] < \tau < d$. 
Abbreviations and Notations

MLD — maximum-likelihood decoding (decoder)
$q$SC — $q$-ary symmetric channel
GF($q$) — Galois Field of size $q$
MDS — maximum-distance separable
RS — Reed-Solomon
KE — key equation
EKE — extended key equation
FIA — fundamental iterative algorithm
GS — Guruswami-Sudan
BIBD — balanced incomplete block design
QBBBD — quasi balanced incomplete block design
d$_H(x, y)$ — Hamming distance between $x$ and $y$
w$_H(x)$ — Hamming weight of $x$
\text{deg } f(x) — degree of univariate polynomial $f(x)$
\text{lead}(Q(x, y)) — leading coefficient of bivariate polynomial $Q(x, y)$
$F_k[x]$ — ring of polynomials of degree less than $k$ over $F$
$\mathbb{Z}_m$ — ring of integers modulo $m$
$\Delta_t(n, d)$ — maximal list-$\ell$ decoding radius of worst $(n, M, d)$ code.
$\Delta^\text{RS}_t(n, d)$ — maximal list-$\ell$ decoding radius of worst $[n, k, d]$ RS code.
$[\tau_t(n, d)] - 1$ — list-$\ell$ decoding radius for $[n, k, d]$ RS codes in Guruswami-Sudan’s algorithm
$\mathcal{C}$ — code
$\mathcal{D}$ — decoder
$n$ — code length
$k$ — code dimension
$d$ — minimum Hamming distance
$\ell$ — maximal list size
$\tau$ — decoding radius
$\alpha_1, \alpha_2, \ldots, \alpha_n$ — code locators
$v$ — received word
$Q(x, y)$ — bivariate polynomial in Sudan’s and Guruswami-Sudan’s algorithms
$Q^{(t)}(x)$ — coefficient of $y^t$ in $Q(x, y)$
$N_t$ — maximal number of nonzero coefficients in $Q^{(t)}(x)$
$\mathcal{L}$ — failing list
$\mathcal{P}$ — partition vector
$\rho_r(\ell)$ — the ratio $(r(r-1))/(\ell(\ell+1))$
$\bar{N}$ — Average number of incorrect codewords in decoding list
Chapter 1

Introduction

1.1 Error correcting codes and their decoding

The material summarized in this section may be found in many books on error-correcting codes, such as [27], [7], [29], and [32].

1.1.1 Using block codes through discrete memoryless channels

Error correcting codes are used to detect and correct errors that fall in a data when transmitted through noisy channels. The channels considered are usually communication channels or storage devices.

A probabilistic model is used to describe the channel. We consider here the model of a discrete memoryless channel (DMC). The transmitted words and the received words are defined over some finite alphabet $F$ (we assume here that the input alphabet and the output alphabet of the channel are identical), and a conditional probability distribution $\text{Prob}(Y | X)$ is defined, where $X$ and $Y$ are discrete random variables that take the alphabet symbols as their values. If $x, y \in F^n$, where $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$, then the probability that $y$ is received through the channel, given that $x$ is transmitted, is denoted here by $\text{Prob}(y | x)$ and is defined by

$$\text{Prob}(y | x) = \prod_{i=1}^{n} \text{Prob}(Y = y_i | X = x_i).$$

An $(n, M)$ block code over an alphabet $F$ is an $M$-subset of $F^n$. The parameter $n$ is referred to as the code length, the number $M$ of codewords in $C$ is called the code size, and the ratio $[\log_{|F|} M] / n$ is considered the code rate. In order to handle errors, block codes are used in an encoding-decoding scheme as described below.

The original data is divided into information words of length $[\log_{|F|} M]$, and each information word is encoded into some codeword $c \in C$ through a one-to-one encoder mapping. The codewords generated by the encoder are transmitted through the channel. We will assume that each codeword is transmitted with the same probability (meaning that $\log_{|F|} M$ is an integer and that each information word occurs with the same probability).

The output of the channel is a sequence of received words in $F^n$. Each received word $y$ is mapped by a decoder mapping $D$ into some codeword $c' \in C$. The codeword $c'$ is then
uniquely associated with an information word. The decoding error probability $P_{\text{err}}$ of $D$ is the probability that a codeword other than $c$ is generated by $D$, given that the respective transmitted codeword is $c$, maximizing over all $c \in \mathcal{C}$.

### 1.1.2 Maximum-likelihood decoding in the $q$-ary symmetric channel

A maximum-likelihood decoder (MLD) maps each word $y$ to a codeword $c'$ that maximizes the probability $\text{Prob}(y | c')$. The maximum likelihood decoding strategy is optimal by means of decoding error probability. We will therefore denote by $P_{\text{err}}(\mathcal{C})$ the error decoding probability of the best MLD for $\mathcal{C}$. (MLDs may defer in their output in cases where the probability $\text{Prob}(y | c')$ is maximized by several codewords $c'$).

When defining the code parameters and selecting the specific codewords of a block code $\mathcal{C}$ for a given channel, a trade-off should be made between making $P_{\text{err}}(\mathcal{C})$ small and keeping the code rate high. By Shannon’s famous coding theorems, there is a sequence of block codes $\{\mathcal{C}_i\}_{i=1}^{\infty}$ with parameters $(n_i, M_i)$ over the binary alphabet such that $n_i/M_i \leq R$ and $P_{\text{err}}(\mathcal{C}_i)$ goes to zero when $i \to \infty$ if and only if $0 \leq R < \text{Cap}$, where $\text{Cap}$ is the capacity function of the channel which depends on its probabilistic distribution. The proof of the ‘if part’ uses random coding.

However, for practical applications, the code length should be kept not too long, and the codewords should be selected so that the encoding and decoding can be done efficiently. Therefore, the code will typically be selected to have the highest rate possible while enabling the use of efficient encoding and decoding algorithms and keeping the error probability below some predetermined upper bound.

In this work, we use as our channel model the memoryless $q$-ary symmetric channel ($q$SC). The channel alphabet consist of $q$ symbols and the probability distribution is given by

$$\text{Prob}(Y = y | X = x) = p \quad \text{for every } y \neq x \quad ; \quad \text{Prob}(Y = x | X = x) = 1 - (q-1)p. \quad (1.1)$$

Typically, $q$ is selected to be a prime power and the alphabet symbols are associated with the elements of the finite field $GF(q)$. Such a selection allows for arithmetic operations between the alphabet symbols according to the field definition.

If $x$ and $y$ are the transmitted and the received words, respectively, then the difference vector $y-x$ is considered the error vector. In the $q$SC, all error vectors of the same Hamming weight are equiprobable. Assuming that $p \leq 1/q$, the probability of an error vector decreases when its Hamming weight increases. A maximum likelihood decoder used with the $q$SC defined by (1.1) is thus supposed to come up with a codeword $c$ corresponding to the ‘lightest’ error vector $y-c$.

An $(n, M, d)$ code is an $(n, M)$ code with minimum Hamming distance $d$ between any two distinct codewords. The decoding error probability $P_{\text{err}}(\mathcal{C})$ of an $(n, M, d)$ code $\mathcal{C}$ over $GF(q)$, when used in the $q$SC, is non-increasing with $d$. However, when the code length and the code size are given, the code minimum distance is bounded from above by the Singleton bound (see [8]).

$$d \leq n - \lfloor \log_q M \rfloor + 1. \quad (1.2)$$
1.1.3 Decoding strategies for algebraic codes

To enable efficient encoding and decoding, block codes with algebraic structure are used. The common basic algebraic requirement is that the \((n, M, d)\) code be a linear subspace of \(F^n\). To emphasize the linearity, the code parameters are denoted by \([n, k]\) or \([n, k, d]\), where \(k = \log_q M\) is the dimension of the vector space \(C\) over \(F\). Encoding information words into codewords of \(C\) can be done efficiently using a generator matrix the rows of which form a basis of \(C\) over \(F\).

NP-completeness of MLD

However, it is shown by Berlekamp, McEliece, and Van Tilborg in [5] that the maximum likelihood decoding problem for a general binary linear code is an NP-complete problem, meaning that an efficient algorithm for solving this problem in a time complexity that can be expressed as a polynomial in the code length \(n\) is not likely to exist.

It should be emphasized though that [5] considers a universal decoder for which the code \(C\) is given as a part of the input. In [10], Bruck and Naor show that maximum likelihood decoding of linear codes remains hard even if the code is known to the decoder in advance and the amount of preprocessing allowed before receiving a word from the channel is not limited. A family of linear block codes is defined for which the existence of a polynomial time decoding algorithm, working under the above conditions, implies that the polynomial hierarchy collapses.

In addition to the above hardness results, it should be mentioned that no polynomial time MLD algorithms are known for any specific family of useful algebraic codes, such as those described in Section 1.3. However, there are well-known efficient algorithms that realize sub-optimal decoding strategies, as described, for example, in Section 1.3. New efficient techniques for decoding algebraic codes are presented in this work.

Decoding up to half the minimum-distance

The Hamming spheres of radius \(\tau\) around the codewords of an \((n, M, d)\) code \(C\) are all disjoint if and only if \(\tau \leq \lfloor (d-1)/2 \rfloor\). In other words, the code \(C\) can recover uniquely and correctly any pattern of \(\tau\) errors or less if and only if \(\tau \leq \lfloor (d-1)/2 \rfloor\). Most of the ‘classical’ decoding algorithms for error-correcting block codes thus realize the strategy of decoding up to \(\lfloor (d-1)/2 \rfloor\) errors, which means providing the only codeword, if any, that differs from \(y\) in \(\lfloor (d-1)/2 \rfloor\) components or less, or declaring (‘detecting’) an error if no such codeword exists. It follows that when more than \(\lfloor (d-1)/2 \rfloor\) errors occur, the decoder either declares an error or gives as output a codeword different from the transmitted one. This decoding strategy is also called bounded-distance decoding and will be referred to in this work as classical decoding.

List decoding

A third strategy, in which we are mainly interested in this work, is list decoding, which was originally considered by Elias and Wozencraft in 1957–1958 (but see [12]) and by Forney [17]. Given the received word \(y\), the task of the decoder is to provide the list of all codewords
that differ from $y$ in $\tau$ components or less, where $\tau$ is a predetermined and fixed parameter which may be greater than $\lfloor(d-1)/2\rfloor$.

More formally, denote by $d_H(y_1, y_2)$ the Hamming distance between two words $y_1, y_2 \in F^n$. We define a list-$\ell$ decoder with a decoding radius $\tau$ for a code $C \subseteq F^n$ as a mapping $D : F^n \rightarrow 2^{\ell}$ such that (i) $|D(y)| \leq \ell$ for every $y \in F^n$, and (ii) $c \in D(y)$ if and only if $c \in C$ and $d_H(c, y) \leq \tau$.

We refer to the codewords in the output list of the decoder as the consistent codewords. In case where $\tau > \lfloor(d-1)/2\rfloor$, there always exists a received word $y$ for which the number of consistent codewords is greater than 1.

Nevertheless, we say that a list decoder with radius $\tau$ corrects all error patterns of Hamming weight $\tau$ or less. This is because the closest codewords to $y$ are included in the output list whenever $\tau$ errors or less fall during transmission. A maximum likelihood decoding can therefore be carried out by using list decoders with increasing decoding radiiuses, until the list is nonempty. If the first nonempty list consists of several codewords, then one of them is arbitrarily chosen.

### 1.2 MDS codes and Reed-Solomon codes

An $(n, M, d)$ code over $F$ for which the Singleton bound (1.2) is satisfied with equality is called maximum-distance separable (MDS). Such a code thus has the best error decoding capability among all $(n, M)$ codes over $F$. The special combinatorial and algebraic properties of linear MDS codes are described in [27, Chapter 11].

The family of generalized Reed-Solomon (RS) codes [27, Ch. 10], is one of the most-studied families of codes. An $[n, k]$ RS code $C_{RS}$ over $F$ is defined through a list of distinct nonzero elements in $F$, referred to as code locators, and the codewords of $C_{RS}$ are obtained by evaluating polynomials of degree $<k$ at the $n$ locators.

Formally, let $\alpha_1, \alpha_2, \ldots, \alpha_n$ be the distinct nonzero code locators. An RS code of dimension $k$ is defined by:

$$C_{RS} = \{ c = (f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_n)) : f(x) \in F_k[x] \},$$

where $F_k[x]$ stands for the set of all polynomials of degree $<k$ in the indeterminate $x$ over $F$.

Two different polynomials in $F_k[x]$ may coincide in up to $k-1$ points, and thus the minimum Hamming distance of $C_{RS}$ is $d = n-k+1$, implying that RS codes are MDS. For this reason, we sometimes use the notation $[n,k]$ instead of $[n,k,d]$, when referring to parameters of RS codes.

A more general definition allows to multiply the components $c_1, c_2, \ldots, c_n$ of each codeword $c$ by respective column multipliers $\beta_1, \beta_2, \ldots, \beta_n$, which are $n$ nonzero (not necessarily distinct) elements of $F$. In this work, we use the definition 1.3, where the column multipliers are assumed to be ones, but the various algorithms and results naturally apply also to the more general definition.

The decoding process of correcting up to $\tau$ errors can thus be viewed as looking for a polynomial $f(x) \in F_k[x]$ that agrees with the received vector $v$ on at least $n-\tau$ values;
namely, \( f(\alpha_j) = v_j \) for at least \( n - \tau \) indexes \( j \). The codeword to which \( v \) should be decoded can then be recomputed by evaluating \( f(x) \) at the code locators.

RS codes are extensively used in magnetic recording and compact disc applications and in communication applications where high reliability is required, such as satellite communications. Though their length \( n \) cannot exceed the field size \( q \), RS codes can be used in concatenation schemes to produce longer codes over smaller fields. Concatenated codes enable the utilization of reliability information corresponding to the various components of the received words, as demonstrated by Forney’s generalized minimum-distance (GMD) decoding algorithm [16]. RS codes, as well as concatenated codes, are suitable also for the task of correcting bursts of errors. This task is typically required in storage applications.

By selecting the code locators and column multipliers in a certain way, cyclic RS codes, called conventional RS codes, are obtained. Several well-known and widely used families of binary codes are related to RS codes: Given a RS code over a finite field \( F \) which is an extension field of \( GF(2) \), the set of codewords whose components are elements of the binary subfield is called an alternant code [27, Ch. 12]. A binary BCH code [27, Ch. 9] is a code consisting of all the binary codewords in a conventional RS code. Note that every decoding algorithm designed for RS codes immediately applies also to conventional RS and to binary alternant and BCH codes.

### 1.3 Decoding Reed-Solomon codes

#### 1.3.1 Classical decoding

Several efficient RS decoding algorithms are known for correcting up to \( \tau = \lfloor (n-k)/2 \rfloor = \lfloor (d-1)/2 \rfloor \) errors. The Berlekamp-Massey algorithm [4, Section 7.4], [28] and the algorithm of Sugiyama et al. [7, Ch. 7], [46] comprise three steps, as described below. Let \( v = (v_1, v_2, \ldots, v_n) \) be the received word.

**Step D0:** Computing syndrome elements \( S_0, S_1, \ldots, S_{n-k-1} \) from the received word \( v \). The syndrome elements are commonly written in a form of a polynomial \( S(x) = \sum_{i=0}^{n-k-1} S_i x^i \).

**Step D1:** Solving the key equation (in short, KE) of RS codes

\[
\Lambda(x) \cdot S(x) \equiv \Omega(x) \pmod{x^{n-k}}
\]

for the error-locator polynomial \( \Lambda(x) \) of degree \( \leq \tau \) and for the error-evaluator polynomial \( \Omega(x) \) of degree \( < \tau \) \((\leq n-k-\tau)\).

**Step D2:** Locating the errors (through computing the roots of \( \Lambda(x) \)) and finding their values (from \( \Lambda(x) \) and \( \Omega(x) \)).

To specify the time complexity of algorithms, we count operations of elements in \( F \) and use the notation \( h_1(m) = O(h_2(m)) \) to mean that there exist positive constants \( c \) and \( m_0 \) such that \( h_1(m) \leq c \cdot h_2(m) \) for all integers \( m \geq m_0 \) [2, Ch. 1]. Based on techniques presented in [2, Ch. 8], procedures for evaluating polynomials in \( F_n[x] \) at \( n \) points in \( F \), as
well as interpolating such polynomials given their values at \( n \) points in \( F \), can be carried out in time complexity \( O(n \log^2 n) \).

Denote by \([n]\) the set of integers \( \{1, 2, \ldots, n\} \). The syndrome elements in Step D0 are computed by

\[
S_i = \sum_{j=1}^{n} v_j \eta_j a_j^i
\]

(1.4)

where

\[
\eta_j^{-1} = \prod_{r \in [n]\setminus\{j\}} (\alpha_j - \alpha_r).
\]

(1.5)

Using a result by Kaminski et al. in [24], it can be shown that the time complexity of computing (1.4) is the same as that of evaluating a polynomial in \( F_n[x] \) at the code locators \( \alpha_j, j \in [n] \), and is therefore \( O(n \log^2 n) \).

Step D2 can be carried out through Chien search [11] and Forney’s algorithm (see [7]). Both algorithms involve evaluation of polynomials at given points, implying that reconstructing the codewords in Step D2 can be executed in time complexity \( O(n \log^2 n) \).

Step D0 is commonly applied while the received word is read into the decoder, and Step D2 is carried out while the correct codeword is flushed out. On the other hand, Step D1 is executed only after the whole received word has been read but before any output is generated; hence, minimizing the complexity of Step D1 means reducing the latency of the decoder. The complexity of Step D1 is discussed in the rest of this section.

Writing \( \Lambda(x) = \sum_{i=0}^{n-k-\tau} \Lambda_i x^i \), we obtain from the KE the following set of \( \tau \) homogeneous equations in the coefficients of \( \Lambda(x) \),

\[
\sum_{s=0}^{n-k-\tau} \Lambda_s \cdot S_{i-s} = 0, \quad n-k-\tau \leq i < n-k,
\]

(1.6)

where \( \Lambda_0 = 1 \). Assuming that \( \tau = (n-k)/2 \) (in the typical case where the code’s minimum distance is odd), the set of equations (1.6) can be expressed in matrix form as

\[
\begin{pmatrix}
S_0 & S_1 & \cdots & S_\tau \\
S_1 & S_2 & \cdots & S_{\tau+1} \\
\vdots & \vdots & \ddots & \vdots \\
S_{\tau-1} & S_\tau & \cdots & S_{2\tau-1}
\end{pmatrix}
\begin{pmatrix}
\Lambda_\tau \\
\Lambda_{\tau-1} \\
\vdots \\
\Lambda_0
\end{pmatrix} = \mathbf{0}.
\]

(1.7)

Massey’s algorithm [28] solves these equations in time complexity \( O(\tau^2) \), and it can be considered as a method for finding the shortest linear shift-register (LSR) that generates a zero sequence when fed with the sequence of syndrome elements. The coefficients of \( \Lambda(x) \) function as the ‘taps’ of the LSR, as illustrated in Figure 1.1. Alternatively, \( \Lambda(x) \) defines a minimal recurrence relation satisfied by the elements \( S_0, S_1, \ldots, S_{2\tau-1} \). An equivalent algorithm has been suggested by Berlekamp and can be found in his book [4].

In [13], an algorithm, referred to as the Fundamental Iterative Algorithm (FIA), is presented for solving general sets of homogeneous linear equations. Given an arbitrary \( \tau \times (\tau+1) \) matrix, the FIA finds the leftmost column in the matrix that is linearly dependent on the
previous columns and gives as output the (nontrivial) vanishing linear combination of those columns. The time complexity of the FIA for an arbitrary $\tau \times (\tau + 1)$ matrix is $O(\tau^3)$ (similarly to Gaussian elimination).

However, the matrix in (1.7) has a special structure called *Hankel form*, meaning that the element in location $(\rho, c)$ is equal to the one in location $(\rho + 1, c - 1)$ for all $1 \leq \rho < \tau$ and $1 < c \leq \tau$. As shown in [13], the time complexity of the FIA, when applied to the matrix in (1.7), can be reduced to $O(\tau^2)$. In fact, the application of the FIA in the case of (1.7) is equivalent to Massey’s algorithm [28].

In his various papers (see, for instance, [40] and [41]), Sakata considers extensions of the task addressed by Massey in [28] and presents algorithms for finding the minimal two-dimensional (or multi-dimensional) recurrence relations that are satisfied by a given two-dimensional (respectively, multi-dimensional) array or by several arrays.

We mention here that acceleration methods allow to reduce the complexity of Massey’s algorithm, namely the complexity of Step D1, to $O((n-k)\log^2(n-k))$ [7].

### 1.3.2 List decoding: Sudan’s and Guruswami-Sudan’s algorithms

In his paper [45], Sudan presents a polynomial time list decoding algorithm for $[n, k]$ RS codes of the form (1.3). For codes of rate not greater than $1/3$, his algorithm corrects more than $[(n-k)/2]$ errors.

The algorithm can be described as a method for computing a polynomial $f(x) \in F_k[x]$ from the set of points $\{(\alpha_j, v_j)\}_{j=1}^n$, while taking into account that some of the values $v_j$ may be erroneous. The computation is done in two steps: in the first step, a nonzero bivariate polynomial $Q(x, y)$ is interpolated through the points $\{(\alpha_j, v_j)\}_{j=1}^n$, where $v = (v_1, v_2, \ldots, v_n)$ is the received word. The degree of $Q(x, y)$ as a polynomial in $y$ is bounded from above by a pre-determined parameter $\ell$. This step is therefore referred to as the *interpolation step*. In the second step, all the linear factors $y - g(x)$ of $Q(x, y)$ are found. The codewords to which $v$ is decoded are computed, through re-encoding, from the polynomials $g(x)$, the number of which cannot exceed $\ell$. The second step is referred to as the *factorization step*.

In [20], Guruswami and Sudan presented a generalization of [45], which enables the efficient correction of more than $(n-k)/2$ errors for every $[n, k]$ RS code at any code rate. The improvement over [45] is achieved by forcing a certain multiplicity $r$ on the various roots $\{(\alpha_j, v_j)\}_{j=1}^n$ of the bivariate polynomial $Q(x, y)$. The value of $r$ depends on the code rate and cannot exceed $\ell$. Specifically, $r^2n = O(\ell^2k)$. Sudan’s original algorithm [45] is derived from the improved algorithm by taking $r=1$. 

![Figure 1.1: Error locator polynomial as a linear shift register.](image-url)
Guruswami-Sudan’s algorithm [20] allows the decoding radius to grow up to \( n - \sqrt{(k-1)n-1} \), while keeping the maximal list size \( \ell \) to be polynomial in the code parameters \( k, n \). Specifically, the list size required in order to work with the maximal radius is \( O(\sqrt{kn^3}) \). However, their algorithm can be used to design list decoders with smaller list size and decoding radius, according to the application requirements. Given the code parameters \( n, k, d = n - k + 1 \) and the maximal list size allowed \( \ell \), the maximal decoding radius guaranteed by their algorithm is given by a mapping \( \tau_\ell(n, d) \), which we define in Section 1.5.1.

The time complexity of any list-\( \ell \) decoder derived from the algorithms [45] and [20] is shown to be polynomial in \( n, k, \ell \). In particular, the execution of the interpolation step requires the solution of \( O(r^2n) \) homogenous linear equations in the unknown coefficients of \( Q(x, y) \). If Gaussian elimination is used as an equation solver in the interpolation step, then the complexity of this step is \( O(r^6n^3) \). In the case of Sudan’s algorithm [20], this time complexity equals \( O(n^3) \).

By providing polynomial-time list decoders with decoding radiuses that go beyond the classical radius \( \left\lfloor (d-1)/2 \right\rfloor \), the work of Sudan and Guruswami is considered a breakthrough in the research of efficient decoding procedures for algebraic error-correcting codes. However, the polynomial degrees in the time complexity expressions presented in [45] and [20] are too high for making these algorithms applicable, which lead to the investigation of efficient procedures for implementing the two main steps of the original algorithms. Such procedures are discussed in Section 1.4.

Guruswami-Sudan’s algorithm naturally applies to conventional RS, alternant, and BCH codes. An extension of [45] that applies to algebraic-geometric (AG) codes was presented by Shokrollahi and Wasserman in [44], while [20] includes not only a decoding algorithm for RS codes, but also its extension to decoding AG codes according to the lines of [44].

The possibility of using Guruswami-Sudan’s algorithm for performing soft-decision list decoding of Reed-Solomon codes has already been indicated in their paper [20]. This can be done by fixing different multiplicities to the different roots of \( Q(x, y) \), according to the reliability level of each point. Recently, Koetter and Vardy presented in [25] efficient soft decision decoding algorithms for RS, BCH, and algebraic-geometric codes, based on [20]. Rules for computing the multiplicities, based on the soft-decision reliability information, are investigated in their work. Their method increases the decoding radius when decoding BCH codes over the binary symmetric channel and may sometimes also increase the radius for RS decoding over the \( q \)-SC. The improvement is achieved by forcing the bivariate polynomial \( Q(x, y) \) to have roots of the form \( (\alpha_j, v_j) \), where \( v = (v_1, v_2, \ldots, v_n) \) is the received word, as well as roots of the form \( (\alpha_j, v'_j) \), where \( v'_j \neq v_j \).

1.4 Efficient procedures for list decoding of Reed-Solomon codes

1.4.1 Results of this work

Viewing the algorithms in [45] and [20], it is intriguing to find their relationship with the classical RS decoding algorithms; specifically, can these algorithms be somehow regarded as an extension of the previously-known RS decoding algorithms, and, if so, can we reduce
the time complexity of the counterpart of Step D1 in these algorithm from cubic in $n$ to quadratic in $n$ or in $n-k$? Can the factorization step be carried out efficiently?

In Chapters 3, 4, and 5, we provide positive answers to those questions. Great parts of the material in these chapters, related to efficient procedures for Sudan’s algorithm [45], were published in [37] after being presented in [36].

**Interpolation step in Sudan’s algorithm**

In Chapter 3, we use Sudan’s algorithm [45] as a basis for developing an extended key equation (in short, EKE) which reduces to the (classical) KE when $\tau = \left\lfloor (n-k)/2 \right\rfloor$. The EKE involves an integer parameter $\ell$ which provides an upper bound on the number of codewords that can be at Hamming distance $\leq \tau$ from any received word; we refer to those codewords as the consistent codewords. Specifically, the EKE takes the form

$$\sum_{t=1}^{\ell} \Lambda^{(t)}(x) \cdot x^{k(t-1)} \cdot S^{(t)}(x) \equiv \Omega(x) \pmod{x^{n-k}},$$

where $S^{(1)}(x), S^{(2)}(x), \ldots, S^{(\ell)}(x)$ are ‘syndrome polynomials’ that can be computed from the received word and $\Lambda^{(1)}(x), \Lambda^{(2)}(x), \ldots, \Lambda^{(\ell)}(x)$, and $\Omega(x)$ are polynomials that satisfy certain degree constraints. The KE is a special case of the EKE when $\ell = 1$. There is a certain and direct connection between the polynomials $\Lambda^{(1)}(x), \Lambda^{(2)}(x), \ldots, \Lambda^{(\ell)}(x)$ and the bivariate polynomial $Q(x, y)$ which is required in Sudan’s algorithm. Therefore, solving the EKE for $\Lambda^{(1)}(x), \Lambda^{(2)}(x), \ldots, \Lambda^{(\ell)}(x)$, and $\Omega(x)$ is equivalent to interpolating $Q(x, y)$ out of the received word.

To compute the polynomials $\{\Lambda^{(t)}(x)\}_{t=1}^{\ell}$, we first translate the EKE into a set of $\tau$ homogeneous linear equations, representing a kind of recurrence relation. In Chapter 5, we demonstrate how this equation set can be derived directly from Sudan’s equations, without going through the EKE. The transition between Sudan’s equations and our equations requires the computation of the syndrome polynomials, and can be done in time complexity $O(\ell n \log^2 n)$. This computation is the counterpart of Step D0 in classical decoding algorithms.

To solve the new set of homogeneous linear equations, we provide two quadratic time algorithms. The first algorithm is a direct application of the FIA which takes advantage of the special ‘multi-Hankel’ matrix form in which our homogenous equations can be written. The validity of this algorithm is implied by the validity of the FIA.

For example, when $\ell = 2$, our algorithm finds the shortest linear dependency of columns in a matrix of the form (1.8). This matrix can be seen as obtained by interleaveing the columns of two different Hankel matrices that have the same number of rows but different number of columns. The interleaving therefore starts only after the $k$th column. The elements in the larger Hankel matrix are the coefficients of the syndrome polynomial $S^{(1)}(x)$, while the elements in the smaller matrix are the coefficients of $S^{(2)}(x)$.

$$\begin{pmatrix}
S^{(1)}_0 & S^{(1)}_1 & \cdots & S^{(1)}_{k-2} & S^{(1)}_{k-1} & S^{(2)}_0 & \cdots & S^{(2)}_{N_1-1} & S^{(2)}_{N_1-2} \\
S^{(1)}_1 & S^{(1)}_2 & \cdots & S^{(1)}_{k-1} & S^{(2)}_1 & S^{(2)}_0 & \cdots & S^{(2)}_{N_1} & S^{(2)}_{N_2} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
S^{(1)}_{\tau-1} & S^{(1)}_{\tau} & \cdots & S^{(1)}_{\tau+k-3} & S^{(2)}_{\tau-1} & S^{(2)}_{\tau-k-2} & \cdots & S^{(2)}_{\tau+N_1-1} & S^{(2)}_{\tau+N_1-2}
\end{pmatrix}.$$

(1.8)
The second algorithm, which we consider as the improved one, is based on the approach of Massey [28] and Sakata [40], as presented in [34]. Even though our algorithm bears similarity to Sakata’s algorithm—and both Sakata’s algorithm and ours generalize Massey’s algorithm—the two generalizations are not the same. The correctness proof of the improved algorithm is based on the correctness proof given in [13] for the FIA. However, the improved algorithm is not a direct application of the FIA.

The time complexity of both algorithms is $O(\ell \tau^2)$, where $\tau$ is the maximal number of errors that Sudan’s algorithm is designed to correct with a list of up to $\ell$ codewords. The value of $\tau$ is smaller than the code’s minimum Hamming distance, and thus the time complexity attained here is $O((n-k)^2)$. As for space complexity, the improved algorithm allocates about $(\ell+1)\tau$ variables that take values in $F$. Up to $\ell\tau$ variables are used to express the output, where the other $\tau$ are used for internal computations of the algorithm.

As was mentioned, solving the EKE directly leads to finding a bivariate polynomial $Q(x, y)$ as required in Sudan’s algorithm. The task of efficiently interpolating a bivariate polynomial through the points $(\alpha_j, \nu_j)_{j=1}^n$ is thus addressed in Chapter 3. In Guruswami-Sudan’s algorithm, the construction of $Q(x, y)$ out of the $n$ points is not equivalent to a simple interpolation, as explained in Section 1.3.2. In Chapter 5, we discuss the way of extending our approach from Chapter 3 to attain a quadratic time procedure for computing $Q(x, y)$ in cases where the multiplicity parameter $r$ is greater than 1. The time complexity of the suggested algorithm is $O(\ell r^4 n^2)$ and its space complexity is $O(\ell r^2 n)$.

**Finding linear factors**

In the last step of Sudan’s and Guruswami-Sudan’s algorithms, the codewords are reconstructed from the bivariate polynomial $Q(x, y)$ through finding its various linear factors $y-g(x)$. There are known general algorithms for factoring multivariate polynomials [48]; yet, those algorithms have relatively large complexity and are therefore not recommended when the required application is finding only the linear factors.

In Chapter 4, we present an efficient procedure for computing the linear factors of a given general bivariate polynomial $Q(x, y)$. If the degree of $Q(x, y)$ in the indeterminate $y$ is $\ell$, then root-finders of degree-$\ell$ univariate polynomials over $F$ lay in the basis of our procedure. The linear factors can be found through our procedure in time complexity $O((\ell \log^2 \ell) k (r^2 n + \ell \log q))$ (where $r$ is the multiplicity parameter that takes values between 1 and $\ell$).

**Summary**

In Section 5.1, we provide a summarizing example of the two-steps quadratic time list decoding algorithm derived from Sudan’s algorithm [45]. The data presented in the example was generated by a computer program implementation of the efficient procedures.

We emphasize that the contribution of the procedures presented in Chapters 3–4 goes beyond the task of efficient list decoding. An efficient method for interpolating a bivariate polynomial from given points is developed in Chapter 3, and an efficient method for finding the linear factors of a given bivariate polynomial is demonstrated in Chapter 4.
1.4.2 Related works

Several other works have been presented since the publication of Sudan’s paper [45], providing efficient procedures for Sudan’s and Guruswami-Sudan’s algorithms for decoding RS codes as well as algebraic-geometric (AG) codes.

Interpolation Step

Each of the various procedures provided for this step exploits in a different way the special structure of the original homogenous linear equations in Sudan’s and Guruswami-Sudan’s algorithms. This way, quadratic time algorithms are derived instead of cubic time Gaussian elimination.

Nielsen and Hoholdt present in [31] a quadratic time procedure for interpolating $Q(x, y)$ which directly solves the original equations of Sudan and Guruswami-Sudan’s algorithms without preprocessing them. The time complexity of their procedure is $O(\ell r^5 n^2)$, where $r$ is the multiplicity of the various $n$ roots of $Q(x, y)$, and the space complexity is $O(\ell r^3 n)$. In [22], the authors generalize their method for decoding Hermitian codes.

A recent work [14] [15] by Feng presents the method in [31] as an application of the FIA to Guruswami-Sudan’s equations which takes advantage of their special structure in order to reduce the number of operations. An outline for accelerating the method of Nielsen and Hoholdt through a recursive divide-and-conquer approach is drawn as well. For RS decoding, the runtime is claimed to be $O(n \log^3 n)$, when multipliers depending on the list size are ignored.

Olshevsky and Shokrollahi [35] solve the equations in Sudan’s and Guruswami-Sudan’s algorithms using an algebraic tool called ‘the displacement operator’. The time complexity of their solution is $O(\ell n^2)$ for Sudan’s equations and $O(\ell r^4 \log_q rn^2)$ for Guruswami-Sudan’s equations.

In the work of Sakata et al. [42], as in our work, the original equations are first preprocessed to yield another set of homogenous linear equations that define a certain recurrence relation. Though the preprocessing methods are different, both reduce to the Berlekamp-Massey syndrome computation when the maximal list size is 1. The preprocessing in [42] implies that the bivariate polynomial $Q(x, y)$ corresponds to a linear recurrence relation which simultaneously defines $r$ different two-dimensional arrays. Consequently, $Q(x, y)$ can be found using Sakata’s algorithm [41] in time complexity $O(\ell r^4 n^2)$. To solve Sudan’s equations, $O(\ell n^2)$ operations are required, and the algorithm reduces to Sakata’s algorithm in [40]. The space complexity of the solutions is not indicated, but the algorithm in [40], for instance, keeps about $2\ell n$ variables over $GF(q)$. In his recent work [43], Sakata presents a fast method to solve the interpolation step in the Guruswami-Sudan’s algorithm for one-point AG codes.

Finding linear factors

Several procedures, rather than full factorization, have been suggested for the task of finding the linear factors $(y - f(x))$ of the bivariate polynomial $Q(x, y)$ over $F$. In [31], Nielsen and Hoholdt translate this problem into the one of finding the roots of a univariate polynomial $Q(y)$ over the finite field $GF(q^k)$ (recall that $k - 1$ is the maximal degree possible for
known root-finding methods, such as Rabin’s probabilistic algorithm [33], can then be applied in order to find the required roots.

More efficient algorithms, different from our algorithm, were presented by Gao and Shokrollahi in [18] and by Augot and Pecquet in [1]. Their works also treat the factorization step in Guruswami-Sudan’s algorithm for AG codes. For Reed-Solomon codes, the time complexities indicated for these two algorithms are $O(\ell^3 k^2)$ and $(n^2 \log n)$, respectively.

The algorithm presented in Chapter 4 can be considered a method for finding roots of univariate polynomials over polynomial rings. Our algorithm was extended by Wu and Siegel in [47] into an efficient method for finding the roots of univariate polynomials over function fields. This method is used for the second step of decoding AG codes.

A way to accelerate the algorithm of Chapter 4 is indicated in Feng’s work [14]. The acceleration is obtained through a recursive divide-and-conquer approach, and is claimed to have time complexity $O(n \log^2 n)$.

### 1.5 Bounds on the list-decoding radius

Denote by $\Delta_\ell(C)$ the largest decoding radius of any list-$\ell$ decoder for a code $C \subseteq F^n$. The value $\Delta_\ell(C)$ is the largest integer value $R$ such that all Hamming spheres of radius $R$ in $F^n$ contain at most $\ell$ codewords of $C$.

Hereafter, by an admissible quadruple $(\ell, n, d, q)$ we mean that $\ell, n, d,$ and $q$ are positive integers such that $1 \leq d \leq n$. By an RS-admissible quadruple $(\ell, n, d, q)$ we mean an admissible quadruple for which, in addition, $n \leq q$ and $q$ is a power of a prime.

Given an admissible quadruple $(\ell, n, d, q)$, we define

$$\Delta_\ell(n,d;q) = \min_C \Delta_\ell(C),$$

where the minimum is taken over all $(n, M, d)$ block codes over an alphabet of size $q$. For an RS admissible quadruple $(\ell, n, d, q)$, we also define

$$\Delta^{\text{RS}}_\ell(n,d;q) = \min_C \Delta_\ell(C),$$

where the minimum is taken over all $[n,k,d]$ RS codes over $GF(q)$. Studying these two quantities is the subject of Chapters 6–7.

Taking the minimum in (1.9) or (1.10) results in the value $\Delta_\ell(C)$ of the ‘worst’ code $C$ in the respective family. In particular, we are interested here in the attainable performance of list-$\ell$ decoders of RS codes (i.e., in the largest number of errors that can be corrected by such decoders), independently of the particular choice of the code locators. From the practical side, this is justified by the structure of existing RS decoding algorithms, such as those described in Sections 1.3 and 1.4, which are typically not tailored to specific selection of code locators.

Clearly, the quantities $\Delta_\ell(n,d;q^m)$ and $\Delta^{\text{RS}}_\ell(n,d;q^m)$ are non-decreasing with $\ell$ and non-increasing with $m$, and for every admissible quadruple $(\ell, n, d, q)$,

$$\Delta_\ell(n,d;q) \leq \Delta^{\text{RS}}_\ell(n,d;q).$$
As explained in Section 1.1, it is well-known that

\[ \Delta_1(n, d; q) = \Delta_1^{\text{RS}}(n, d; q) = \left(\frac{d-1}{2}\right) \]

(independently of \( q \)).

### 1.5.1 Previous work

The decoding radius guaranteed by Guruswami-Sudan’s algorithm [20] is naturally a lower bound on \( \Delta_1^{\text{RS}}(n, d; q) \). The exact decoding radius achieved by [20] is therefore referred to here a the GS bound and is defined in Theorem 1.1 below. The presentation here is different from the one in the original paper [20].

We first introduce several notations that are required not only for the statement of Theorem 1.1, but also in our analysis throughout Chapters 6-7.

Given \( \ell \geq 1 \), partition the real interval \([0, 1)\) into the \( \ell \) sub-intervals

\[ [0, \rho_2), [\rho_2, \rho_3), \ldots, [\rho_{\ell}, 1), \]

where

\[ \rho_r = \rho_r(\ell) = \frac{r(r-1)}{\ell(\ell+1)}, \quad r = 1, 2, \ldots, \ell + 1. \]

(1.12)

Given integers \( n \) and \( d \) such that \( 1 \leq d \leq n \), define the relative minimum distance \( \delta = d/n \). Let \( r = r(\delta) \) be the unique index such that \( 1 - \delta \in [\rho_r, \rho_{r+1}) \). Also define

\[ \tau_{\ell}(n, d) = \frac{1}{(\ell+1)^{\ell+1}} \left( \frac{\ell+1}{2} \right)^{\ell+1} d - \left( \frac{\ell+1}{2} \right) n \].

(1.13)

The mapping \( \delta \mapsto \tau_{\ell}(n, \delta) \) is piecewise-linear and continuous over \([0, 1)\) for every fixed \( n \). It can be easily verified that \( \tau_{\ell}(n, d) < d \), for every value of \( \ell \), assuming \( d \leq n \). One can also verify that when \( 1 - \delta \geq \rho_r \),

\[ \tau_{\ell}(n, d) = d/2. \]

By its definition, \( \tau_{\ell}(n, d) \) is an integer if and only if

\[ (\ell+1)^{\ell+1} \text{ divides } \left( \frac{\ell+1}{2} \right)^{\ell+1} d - \left( \frac{\ell+1}{2} \right) n. \]

(1.14)

**Theorem 1.1**  For every RS-admissible quadruple \((\ell, n, d, q)\), the list-\( \ell \) decoder in [20] for an \([n, k, d]\) RS code over \(\text{GF}(q)\) has a decoding radius \([\tau_{\ell}(n, d)] - 1\); so,

\[ \Delta_{\ell}^{\text{RS}}(n, d; q) \geq [\tau_{\ell}(n, d)] - 1. \]

(1.15)

The proof of Theorem 1.1, namely the derivation of the GS bound from the algorithm presentation in [20], is included in Chapter 2. We comment that the decoding radius indicated in Theorem 1.1 is achieved when the parameter \( r \), selected so that \( 1 - d/n \in [\rho_r, \rho_{r+1}) \), is taken to be the *multiplicity parameter* in Guruswami-Sudan’s algorithm, as explained in Section 1.3.2. By the definition (1.12) of \( \rho_r \) it thus follows that \( r^2 n = O(\ell^2 k) \).

In a recent work [23], Justesen and Høholdt compute RS-admissible quadruples \((\ell, n, d, q)\) for which there exist \((n, q^{n-d+1}, d)\) MDS and RS codes over \( F = \text{GF}(q) \) that attain the GS bound. A key ingredient in their technique is constructing what we call here a *failing list* of codewords. By a failing list of size \( \ell+1 \), we mean a set of \( \ell+1 \) words, \( \{c_0, c_1, \ldots, c_\ell\} \subseteq F^n \), such that the following two conditions hold:
\textbullet \ d_H(c_s, c_t) \geq d \text{ for every } 0 \leq s < t \leq \ell, \text{ and} \\
\textbullet \text{there is some } v \in F^n \text{ such that } d_H(c_s, v) \leq \lceil \tau_\ell(n, d) \rceil \text{ for every } 0 \leq s \leq \ell.

One can easily see that a failing list of size \( \ell+1 \) is contained in an \((n, M, d)\) code \( C \) if and only if \( \Delta_\ell(C) \) attains the GS bound. Several families of MDS codes and RS codes that contain such failing lists are presented in [23]; their constructions are based on block designs, and in each of these constructions, the relative minimum distance \( \delta \) is such that \( 1-\delta = \rho_\ell(\ell) \).

We point out that bounds on \( \Delta_\ell^{\text{RS}}(n, d) \) naturally imply bounds on the maximal list size which is actually realized by the best list decoder with a given radius \( \tau \) for an \([n, k, d]\) RS code. As mentioned in Section 1.3.2, Guruswami-Sudan's algorithm proves that for \( \tau \approx n - \sqrt{(k-1)n} = n(1 - \sqrt{1-d/n}) \), the maximal list size is polynomial in the code length \( n \). However, Justesen and Hoholdt show in [23] that the maximal list size becomes exponential in \( n \) when \( \tau \) is very close to \( d \). Their result holds for all \([n, k, d]\) MDS codes, and not only for RS codes. An interesting open problem is to describe the behavior of the list size for RS (MDS) decoding radiuses in the range between \( n-\sqrt{k}n \) and \( n-k \).

### 1.5.2 Results of this work

The results presented in this section and in Chapters 6–7 have been partially presented in [38] and have been submitted for publication in \textit{SIAM J. Disc. Math.}

In Chapter 6, we first generalize Theorem 1.1 by showing that the GS bound is a lower bound on the list-\( \ell \) decoding radius of every \((n, M, d)\) block code over any finite alphabet.

**Theorem 1.2** Let \((\ell, n, d, q)\) be an admissible quadruple. Then

\[
\Delta_\ell(n, d; q) \geq \lceil \tau_\ell(n, d) \rceil - 1 ,
\]

where \( \tau_\ell(n, d) \) is defined by (1.13).

The proof of Theorem 1.2 is obtained by using combinatorial arguments, while the result in Theorem 1.1 is based on algebraic analysis.

Note that in the particular case where \( d/n \leq 2/(\ell+1) \), the GS bound becomes the ‘classical’ lower bound on the list-1 decoding radius:

\[
\Delta_1(n, d; q) \geq \lceil (d-1)/2 \rceil = \Delta_1(n, d; q) .
\]

In Sections 6.2–6.3, we turn to investigate the conditions under which the GS bound is tight (or not tight), namely the conditions for the existence (resp., non-existence) of failing lists in a given block code. We introduce a combinatorial configuration, akin to block designs, that defines a structure of failing lists which covers the whole range of rational \( \delta \) values (and not just those for which \( 1-\delta = \rho_\ell(\ell) \)).

Furthermore, we prove that for triples \((\ell, n, d)\) for which \( \tau_\ell(n, d) \) is an integer, our structure completely characterizes the failing lists of size \( \ell+1 \) in any given \((n, M, d)\) code over any alphabet \( F \). This, in turn, provides sufficient and necessary conditions on the existence of such failing lists (see Proposition 6.3 in Section 6.2).
Some of the necessary conditions are related to the existence of some well-defined combinatorial structures, such as constant-weight codes and balanced incomplete block designs (BIBDs). In cases where such structures are known not to exist for the given parameters, we conclude that a failing list does not exist either and the lower bound on $\Delta_L(n, d; q)$ can thus be improved (by at least 1) over the GS bound.

In Chapter 7, our discussion moves to RS codes and, specifically, to the derivation of bounds on $\Delta_{RS}^L(n, d; q)$. Here, we use the identity $k-1 = n-d$, and we slightly modify the common definition of rate of an $[n, k, d]$ MDS code and use it for the value $(k-1)/n = 1-d/n$; as it turns out, this value fits more conveniently into our analysis.

First, we obtain sufficient and necessary conditions for the existence of failing lists in RS codes (see Lemma 7.7 in Section 7.2). Using our sufficient conditions, we identify families of RS codes (other than those obtained in [23]) that attain the GS bound.

For triples $(\ell, n, k)$ that correspond to the high-rate and low-rate sub-intervals (specifically, $(k-1)/n \leq \rho_2 = 2/(\ell(\ell+1))$ or $(k-1)/n \geq \rho_6 = 1 - (2/(\ell+1)))$, we find a variety of finite fields $GF(q)$ over which there are $[n, k, d]$ RS codes that attain the GS bound. For the high-rate case (i.e., small values of $d/n$), we identify quadruples $(\ell, n, d, q)$ for which a list-$\ell$ decoder for the worst $[n, k, d]$ RS code, and hence for the worst $(n, M, d)$ code, does no better than a list-1 (‘classical’) decoder.

As for the intermediate sub-intervals, namely

$$\frac{2}{\ell+1} < \frac{d}{n} < 1 - \frac{2}{\ell(\ell+1)};$$

this range is nonempty for $\ell \geq 3$. The treatment of this range seems to be more elaborate than the extreme (rightmost and leftmost) sub-intervals. Hence, our results for the mid-rate range are quite partial; yet, they demonstrate that the techniques that are developed in Chapters 6–7 are applicable not only to the extreme sub-intervals.

On the other hand, as part of our treatment of the mid-rate range, we also find RS-admissible quadruples $(\ell, n, d, q)$ for which the GS lower bound is not tight. Such an improvement on the GS bound sometimes follows simply from the nonexistence of certain combinatorial structures, in which case the improvement applies to every $(n, M, d)$ code. However, there are cases where stronger results can be obtained specifically for RS codes. Two such examples are provided where we show that any failing list cannot be realized by RS codes, namely by evaluating polynomials at certain points.

Finally, in Chapter 8, our investigation moves from bounding the maximal list size of a decoder with a given radius $\tau$ to bounding the average number of codewords included in the output list. The average is taken over all transmitted codewords and error vectors, where the probabilistic model assumed is that of the $q$-ary symmetric channel. The discussion applies to MDS codes and takes advantage of their special structure.

### 1.6 Organization of this work

Some technical background is included in Chapter 2: The details of Sudan’s and Guruswami-Sudan’s algorithms are presented, and the GS bound on $\Delta_{RS}^L(n, d)$, that was claimed in Theorem 1.1, is proved. In addition, the FIA defined in [13] is described.
In Chapter 3, we present an efficient implementation of the first decoding step in Sudan’s algorithm by providing a quadratic time procedure for interpolating a bivariate polynomial from given points. In Chapter 4, an efficient implementation of the second decoding step is provided by the definition of a quadratic time procedure for finding the linear factors \( y - g(x) \) of a given bivariate polynomial. In Chapter 5, the efficient implementation of Sudan’s algorithm is summarized and its extension into a fast implementation of Guruswami-Sudan’s algorithm is outlined.

In Chapter 6, lower bounds are derived on the list decoding radius \( \Delta_l(n, d) \) of general block codes. The combinatorial structure of failing lists is characterized. In Chapter 7, lower bounds and upper bounds are derived on the list decoding radius \( \Delta^R_S(n, d) \) of RS codes. Upper bounds on the average number of incorrect codewords in list decoding of MDS codes are derived in Chapter 8.

The conclusions from this work, as well as some open problems that require further research, are described in Chapter 9.
Chapter 2

Technical background

2.1 Sudan’s algorithm

Let $C$ be any $[n, k]$ RS code over a finite field $F$. Given a prescribed upper bound $\ell$ on the number of consistent codewords, Sudan’s algorithm \cite{45} corrects any error pattern of up to $\tau$ errors for

$$\tau = n - (m + 1) - \ell(k - 1),$$

where $m$ is any nonnegative integer satisfying

$$(m + 1)(\ell + 1) + (k - 1)\left(\frac{\ell + 1}{2}\right) > n. \quad (2.2)$$

Define

$$N_t = m + 1 + (\ell - t)(k - 1). \quad (2.3)$$

Sudan’s algorithm is based on the following two lemmas, taken from \cite{45}.

**Lemma 2.1** Whenever (2.2) holds, there exists a nonzero bivariate polynomial $Q(x, y)$ over $F$ that satisfies

$$Q(x, y) = \sum_{t=0}^{\ell} Q^{(t)}(x)y^t, \quad \deg Q^{(t)}(x) < N_t, \quad (2.4)$$

and

$$Q(\alpha_j, v_j) = 0, \quad \forall j \in [n]. \quad (2.5)$$

A set of homogenous linear equations in the coefficients of $Q(x, y) = \sum_{t=0}^{\ell} \sum_{s=0}^{N_t-1} Q^{(t)}_s x^s y^t$, which is equivalent to the requirement (2.5), is given by

$$\sum_{t=0}^{\ell} \sum_{s=0}^{N_t-1} Q^{(t)}_s \alpha_j^s v_j^t = 0, \quad \forall j \in [n]. \quad (2.6)$$

**Lemma 2.2** Suppose that (2.1)–(2.2) hold. Let $Q(x, y)$ be a nonzero bivariate polynomial over $F$ satisfying (2.4)–(2.5) and let $f(x) \in F_k[x]$ be such that $f(\alpha_j) = v_j$ for at least $n - \tau$ locators $\alpha_j$. Then $Q(x, f(x))$ is identically zero.

Sudan’s algorithm consists of the following two steps:
**Input:** received word \( v = (v_1, v_2, \ldots, v_n) \).

**Step S1:** Find a nonzero bivariate polynomial \( Q(x, y) \) over \( F \) satisfying (2.4)–(2.5).

**Step S2:** Output all the polynomials \( g(x) \in F_k[x] \) such that

- \( y - g(x) \) is a factor of \( Q(x, y) \), and
- \( g(\alpha_j) = v_j \) for at least \( n - \tau \) locators \( \alpha_j \).

The following parameter analysis is later used in Chapters 3–4. We will further assume that \( \ell, k, \) and \( n \) satisfy the inequality

\[
\ell + (k-1)\binom{\ell+1}{2} \leq n. \tag{2.7}
\]

Indeed, it can be verified that if (2.7) did not hold, then, for the same values of \( k, n, \) and \( \tau \), (2.1) and (2.2) would still be satisfied if we decreased \( \ell \) by 1 and chose \( m = k-1 \); this means that, for the given \( k, n \) and \( \tau \), we could assume a shorter list of consistent codewords. In particular, setting \( \ell = 2 \) in (2.7) implies \( k \leq (n+1)/3 \). Note that from (2.2) and the minimality of \( m \) we obtain

\[
(m + 1)(\ell + 1) + (k-1)\binom{\ell+1}{2} \leq n + \ell + 1, \tag{2.8}
\]

and by (2.1), (2.2), and (2.8),

\[
\tau + 1 \leq \sum_{t=1}^{\ell} N_t \leq \tau + \ell + 1. \tag{2.9}
\]

If we now fix \( k, n, \) and \( \ell \), then from (2.1) it follows that \( \tau \) will be maximized if we choose the smallest possible \( m \) that satisfies (2.2). For \( \ell = 1 \) (the classical case) we thus obtain \( m = \lfloor (n-k)/2 \rfloor \) and \( \tau = n - k - m = \lfloor (n-k)/2 \rfloor \). For \( \ell = 2 \) we obtain \( m = \lceil (n+1)/3 \rceil - k \) (assuming \( k \leq (n+1)/3 \), in conjunction with (2.7)) and \( \tau = n+1 - 2k - m = \lfloor 2(n+1)/3 \rfloor - k \).

### 2.2 Guruswami-Sudan’s algorithm

In Guruswami-Sudan’s algorithm [20], an additional parameter \( r \), which is a nonnegative integer not greater than \( \ell \), is involved. The algorithm now corrects any error pattern of up to \( \tau \) errors if

\[
r \tau \leq rn - (m + 1) - \ell(k - 1), \tag{2.10}
\]

where \( m \) is a nonnegative integer satisfying

\[
(m + 1)(\ell + 1) + (k-1)\binom{\ell+1}{2} > \binom{r+1}{2}n. \tag{2.11}
\]

Note that for \( r = 1 \), the parameter definitions (2.1)–(2.2) satisfy (2.10)–(2.11).

The extensions of Lemmas 2.1–2.2 are stated in the following two lemmas.
**Lemma 2.3** Whenever (2.11) holds, there exists a nonzero bivariate polynomial \( Q(x, y) \) over \( F \) that satisfies

\[
Q(x, y) = \sum_{t=0}^{\ell} Q^{(t)}(x)y^t, \quad \deg Q^{(t)}(x) < N_t,
\]

and the coefficients of total degree less than \( r \) in \( Q(x - \alpha_j, y - v_j) \) are all zero for every \( j \in [n] \). Formally,

\[
\sum_{t=0}^{\ell} \sum_{s=0}^{N_t-1} \binom{s}{s'} \binom{\ell}{t} Q_s(x)\alpha_j^{s-s'}v_j^{t-t'} = 0, \quad t' = 0, \ldots, r - 1, \quad s' = 0, \ldots, r - 1 - t',
\]

where \( j = 1, \ldots, n \).

**Lemma 2.4** Let \( Q(x, y) \) be a nonzero bivariate polynomial over \( F \) satisfying (2.12)–(2.13) and let \( f(x) \in F_k[x] \) be such that \( f(\alpha_j) = v_j \) for at least \( n - \tau \) locators \( \alpha_j \), where \( \tau \) satisfies (2.10). Then \( Q(x, f(x)) \) is identically zero.

The two steps in Guruswami-Sudan’s algorithm are similar to those in Sudan’s algorithm, but the polynomial \( Q(x, y) \) should now satisfy the requirements (2.12)–(2.13) under the parameter definitions (2.10)–(2.11).

### 2.3 Guruswami-Sudan’s decoding radius \( [\tau_\ell(n, d)] - 1 \)

**Proof of Theorem 1.1:** As shown in [20], Guruswami-Sudan’s algorithm defines a list-\( \ell \) decoder with a decoding radius \( \tau \), if there is a nonnegative integer \( m \) such that (2.10)–(2.11) hold. It can be easily verified that every integer \( \tau \) that satisfies (2.10)–(2.11) for some nonnegative integer \( m \) must be smaller than \( \tau_\ell(n, d) \); therefore, \( \tau \leq [\tau_\ell(n, d)] - 1 \). Next we show the converse result: we prove that \( \tau = [\tau_\ell(n, d)] - 1 \) satisfies (2.10)–(2.11) for some positive integer \( m \). Define

\[
m' = \frac{1}{r+1} \left( \binom{r+1}{2} n - \binom{\ell+1}{2} (k-1) \right).
\]

Since \( 1 - d/n = (k-1)/n < \rho_{r+1} \), the value of \( m' \) is positive. We can now incorporate \( m' \) into the expression for \( \tau_\ell(n, d) \) in (1.13) to obtain

\[
\tau_\ell(n, d) = \frac{1}{(r+1)^r} \left( \binom{r+1}{2} (n - (k-1)) - \binom{\ell+1}{2} (k-1) \right) = \frac{1}{r} (rn - m' - \ell(k-1)).
\]

If \( r\tau_\ell(n, d) \) is an integer, then \( m' \) must be an integer too; in this case, \( m = m' \) and \( \tau = \tau_\ell(n, d) - 1 \) satisfy (2.10)–(2.11).

On the other hand, if \( r\tau_\ell(n, d) \) is not an integer, then

\[
r\tau = r([\tau_\ell(n, d)] - 1) < rn - m' - \ell(k-1),
\]

and, therefore,

\[
r\tau \leq rn - [m'] - \ell(k-1).
\]

Hence, \( m = [m'] - 1 \) and \( \tau = [\tau_\ell(n, d)] - 1 \) satisfy (2.10). Furthermore, this value of \( m \) is nonnegative and satisfies (2.11) as well.
Fix the triple \((\ell, n, d)\), and consider the function 
\[
t_{\ell,n,d}(r) = \frac{1}{(\ell+1)r} \left( \binom{\ell+1}{2}d - \left(\frac{\ell+1}{2}\right)n \right).
\]
It can be easily verified that 
\[
t_{\ell,n,d}(r+1) \geq t_{\ell,n,d}(r)
\]
only for \(1 - d/n \geq (r(r+1))/((\ell+1)) = \rho_{r+1}\). Hence, the value of \(r\) which maximizes the decoding radius of Guruswami-Sudan’s algorithm is the one for which \(1 - d/n = (k-1)/n \in [\rho_r, \rho_{r+1}]\), as claimed in Theorem 1.1. \(\blacksquare\)

## 2.4 The FIA

For the sake of completeness, we present in Figure 2.1 (a slightly modified version of) the FIA. The input to the algorithm is a matrix \(S = [S_{\ell,r}]_{\ell=1}^r_{r=0}\) over \(F\). Two arrays, \(A\) and \(D\), are allocated, each indexed by the rows of \(S\). The contents of an entry of \(A\) that corresponds to row \(\rho\) will be a polynomial referred to as \(A[\rho](x)\), and a typical entry \(D[\rho]\) in \(D\) contains a value in \(F\).

The iterations of the main loop of the FIA examine the columns of \(S\) in ascending order, starting with the first column (column 0). The currently-examined entry in \(S\) is pointed at by the row pointer \(\rho\) and the column pointer \(c\). When starting examining column \(c\), an arbitrary polynomial \(T(x)\) of degree (exactly) \(c\) over \(F\) is selected. We comment that the algorithm as it appears in [13] does not give freedom in selecting the polynomial \(T(x)\) with which we start treating column \(c\). However, the validity of the algorithm, as manifested through Lemmas 2.5–2.6 and Theorem 2.7 below, still holds when the polynomial \(T(x)\) is selected arbitrarily each time a new column \(c\) is reached, as long as its degree is \(c\) (without loss of generality we can assume that \(T(x)\) is monic). Each column is examined from top to bottom, starting with its first entry \((\rho = 1)\), but not necessarily reaching its last entry. A typical sequence of values that the pointers \((\rho, c)\) may take throughout the course of the FIA is shown in Figure 2.2.

Each iteration of the main loop of the FIA starts with a computation of the discrepancy \(\Delta\) at the location \((\rho, c)\), using the current value of \(T(x)\). The values of \(\Delta\) and \(D[\rho]\) determine whether line 3 or line 7 in Figure 2.1 will be reached in the current iteration of the main loop. The next two lemmas, taken from [13], specify the dependency properties of the columns in \(S\) when each of those lines is reached.

**Lemma 2.5** Let \(T_{\ell,c}(x) = \sum_{j=0}^{c} T_{j}x^j\) be the value of \(T(x)\) following the execution of line 3 in Figure 2.1 with \((\rho, c) = (\rho, c)\). Then \(T_{\ell,c}(x)\) is a monic polynomial of degree (exactly) \(c\) and \(\sum_{j=0}^{c} T_{j}S_{i,j} = 0\) for all \(i \in [\rho]\).

Note that line 3 can be reached with a pair \((\rho, c) = (\rho, c)\) also when \(\Delta \neq 0\). This can happen only if \(D[\rho]\) is nonzero, in which case \(T(x)\) is updated in line 3. A nonzero value of \(D[\rho]\) indicates that line 7 was reached before with \(\rho = \rho\). Following the execution of line 3, the algorithm either terminates (if \(\rho = \tau\)) or proceeds with the next entry in column \(c\) (if \(\rho < \tau\)).

**Lemma 2.6** Let \(T_{\ell,c}(x) = \sum_{j=0}^{c} T_{j}x^j\) be the value of \(T(x)\) (and \(A[\rho](x)\)) following the execution of line 7 in Figure 2.1 with \((\rho, c) = (\rho, c)\). Then columns 0 through \(c\) in \(S\) are linearly independent and \(\sum_{j=0}^{c} T_{j}S_{i,j} = 0\) for all \(i \in [\rho-1]\).
**Input**: $\tau \times (\tau+1)$ matrix $S = [S_{c,r}]_{c=0}^{\tau-1}_{r=0}$. 

**Data structures**: 
- polynomial $T(x) = \sum_{j=0}^{\tau} T_j x^j \in F_{\tau+1}[x]$; 
- row pointer $\rho \in [\tau]$ and column pointer $\sigma \in \{0, 1, \ldots, \tau\}$; 
- variable $\Delta \in F$; 
- array $A$ of $\tau$ polynomials in $F_{\tau+1}[x]$; 
- array $D$ of $\tau$ entries in $F$. 

**Initialize**: 
for every $\rho \in [\tau]$ do $D[\rho] \leftarrow 0$;  
$\rho \leftarrow 1$; $\sigma \leftarrow 0$; 
$T(x) \leftarrow 1$;

do forever  
\{  
1   $\Delta \leftarrow \sum_{j=0} T_j \cdot S_{\rho,j}$;  
2   if $\Delta = 0$ or $D[\rho] \neq 0$ then  
3       if $\Delta \neq 0$ then $T(x) \leftarrow T(x) - \frac{\Delta}{D[\rho]} \cdot A[\rho](x)$;  
4       if $\rho = \tau$ then  
5           return $T(x)$;  
6       else $\rho \leftarrow \rho + 1$;  
7   else /* $\Delta \neq 0$ and $D[\rho] = 0$ */  
8       $A[\rho](x) \leftarrow T(x)$;  
9       $D[\rho] \leftarrow \Delta$;  
10      $\sigma \leftarrow \sigma + 1$;  
11     $\rho \leftarrow 1$;  
12   $T(x) \leftarrow$ any monic polynomial of degree $\sigma$ over $F$;  
\} 

**Figure 2.1**: Fundamental Iterative Algorithm (FIA).
The thick dots in Figure 2.2 indicate pairs \((\rho, \sigma)\) for which the FIA reaches line 7. Following the execution of lines 7–8, the next examined entry is the first entry of the next column, i.e., the new value of the pair \((\rho, \sigma)\) is \((1, c+1)\).

The following theorem guarantees that the FIA in Figure 2.1 indeed finds the leftmost column in \(S\) that is linearly dependent on the previous columns, and it also computes the respective vanishing linear combination. The proof of the theorem is based on Lemmas 2.5 and 2.6 and can be found in [13].

**Theorem 2.7** The value \(c\) of \(\sigma\) and the polynomial \(T(x) = \sum_{j=0}^{c} T_j x^j\) upon termination of the FIA are such that

(i) \(c\) is the smallest integer for which columns 0 through \(c\) in \(S\) are linearly dependent, and —

(ii) the respective vanishing linear combination of the columns of \(S\) is given by \((T_0, T_1, \ldots, T_c, 0, 0, \ldots, 0)\).

The time complexity of the FIA for an arbitrary \(\tau \times (\tau+1)\) matrix is \(O(\tau^3)\), since the algorithm may need to compute \(O(\tau^2)\) discrepancy values in case columns 0 through \(\tau-1\) are linearly independent. We outline next how the Hankel form, in which \(S_{i,j+1} = S_{i+1,j}\), allows to reduce the complexity.

Suppose a nonzero discrepancy \(\Delta\) is obtained when \((\rho, \sigma) = (\rho, c-1)\). If \(D[\sigma] = 0\) then \(T(x)\) and \(\Delta\) are stored in \(A[\sigma](x)\) and \(D[\sigma]\), respectively, and \((\rho, \sigma)\) is set to \((1, c)\). At this point, any monic polynomial of degree \(\sigma\) over \(F\) can be taken as the new value of \(T(x)\). In particular, we can select \(T(x) = x \cdot A[\sigma](x)\). Writing \(A[\sigma](x) = \sum_{j=0}^{c-1} A_j x^j\), by Lemma 2.6 we have \(\sum_{j=0}^{c-1} A_j S_{i,j} = 0\) for all \(i \in [\sigma-1]\). On the other hand, since the matrix has a Hankel form,

\[
\sum_{j=0}^{c} T_j S_{i,j} = \sum_{j=0}^{c-1} A_j S_{i,j+1} = \sum_{j=0}^{c-1} A_j S_{i+1,j}
\]

for all \(i \in [\tau-1]\). This means that, while at column \(\sigma = c\), the discrepancy computation will result in a zero value for all \(\rho \in [\sigma-2]\). Hence, we can start examining column \(c\) with \(\rho = \sigma-1\) instead of \(\rho = 1\), and the discrepancy value at that point will be \(D[\sigma]\). Figure 2.3 presents a typical sequence of values of \((\rho, \sigma)\) that results from such a short-cut.
Figure 2.3: Typical values of ($\rho, \sigma$) in the FIA when the input has a Hankel form.

Thus, the Hankel form allows to reduce the number of discrepancy computations: no discrepancy needs to be computed above and along the diagonal lines in Figure 2.3. Moreover, it is shown in [13] that when applying the FIA to the matrix in (1.7), a memory of $O(\tau)$ is sufficient and the arrays $A$ and $D$ reduce to scalar variables.
Chapter 3

Efficient interpolation of bivariate polynomials

3.1 Introduction

In this chapter, efficient procedures are suggested for interpolating the bivariate polynomial $Q(x, y)$ in Sudan’s algorithm (see Section 2.1), along the lines drawn in Section 1.4.

We point out that for every bivariate polynomial $Q(x, y) = \sum_{\ell=0}^{t} Q^{(\ell)}(x)y^\ell$ there exist integers $m, k$ such that $\deg Q^{(\ell)}(x) < m+1+(\ell-t)(k-1)$. To see this, define $m = \deg Q^{(\ell)}(x)$, and let $k$ be an integer (say, the smallest integer) such that $\deg Q^{(\ell)}(x) - \deg Q^{(\ell)}(x) \leq (\ell-t)(k-1)$ for every $t < \ell$. It follows that the degree constraint (2.4) is rather general, and the algorithms presented in this section can be thought of as solving the general problem of interpolating a bivariate polynomial through given points.

If the number of interpolation points is smaller than the number of nonzero coefficients allowed, then the existence of a solution is guaranteed. This requirement is satisfied whenever (2.2) holds. There may exist several polynomials that satisfy the degree constraints and go through the given interpolation points. However, the solutions we provide can be considered minimal according criteria that will be described in the sequel.

In Section 3.2, the EKE is derived out of Sudan’s equations (2.4)–(2.5). A formula for computing the syndrome elements is presented. In Section 3.3, a set of linear homogenous equations, which can be seen as a generalization of (1.6), is derived from the EKE. (A derivation of this equation set directly from Sudan’s equations (2.6) is demonstrated in Section 5.3.)

In Section 3.4, an FIA-based algorithm for solving the above equation set is provided. A simpler improved algorithm for solving the same equation set is presented in Section 3.5. Both algorithms compute the polynomials $Q^{(1)}(x), \ldots, Q^{(\ell)}(x)$ defined in (2.4). The computation of $Q^{(0)}(x)$ is considered in Section 3.6.

3.2 Extended key equation

In this section, we derive the EKE based on Sudan’s algorithm. Let $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ be the received word and let $V(x)$ be the unique polynomial in $F_n[x]$ such that $V(\alpha_j) = v_j$
for all \( j \in [n] \); the existence and the uniqueness of the polynomial \( V(x) \) are implied by the Lagrange interpolation theorem [26, Ch. 1].

**Lemma 3.1** Let \( Q(x, y) = \sum_{t=0}^{\ell} Q(t)(x) y^t \) be a bivariate polynomial that satisfies (2.4). Then \( Q(x, y) \) satisfies (2.5) if and only if there exists a polynomial \( B(x) \) over \( F \) for which

\[
\sum_{t=0}^{\ell} Q(t)(x) \cdot (V(x))^t = B(x) \cdot \prod_{j=1}^{n} (x - \alpha_j),
\]

where

\[
\deg B(x) < \ell(n-k) - \tau.
\]

**Proof:** By the definition of \( V(x) \), an equivalent condition to (2.5) is that the univariate polynomial \( Q(x, V(x)) \) vanishes at each of the code locators \( \alpha_j, j \in [n] \). Alternatively, there is a polynomial \( B(x) \) over \( F \) satisfying (3.1). Now, by (2.3) and (2.4),

\[
\deg Q(t)(x) \cdot (V(x))^t < n - \tau + t(n-k), \quad t \in [\ell],
\]

implying that

\[
\deg Q(x, V(x)) < n - \tau + \ell(n-k)
\]

and that \( B(x) \) must satisfy (3.2).

Let \( Q(x, y) \) be a polynomial satisfying (2.4). We will introduce the short-hand bivariate notation \( Q^*(x, y) = \sum_{t=1}^{\ell} Q(t)(x) y^t \). Define the following polynomials obtained by reversing the order of coefficients in \( V(x), \prod_{j=1}^{n} (x - \alpha_j) \), and \( (Q(t)(x))_{t=1}^{\ell} \), respectively:

\[
\begin{align*}
\overline{V}(x) &= x^{n-1} V(x^{-1}), \\
G(x) &= \prod_{j=1}^{n} (1 - \alpha_j x), \\
\Lambda(t)(x) &= x^{n-1} Q(t)(x^{-1}), \quad t \in [\ell].
\end{align*}
\]

**Lemma 3.2** Let \( Q^*(x, y) = \sum_{t=1}^{\ell} Q(t)(x) y^t \) satisfy

\[
\deg Q(t)(x) < N_t, \quad t \in [\ell],
\]

and let \( \overline{V}(x), G(x), \) and \( (\Lambda(t)(x))_{t=1}^{\ell} \) be defined by (3.3). There exists a (unique) polynomial \( Q^{(0)}(x) \) such that \( Q(x, y) = Q^{(0)}(x) + Q^*(x, y) \) satisfies (2.4)-(2.5) if and only if there exists a (unique) polynomial \( \overline{B}(x) \in F_{(n-k)-\tau}[x] \) such that

\[
\sum_{t=1}^{\ell} \Lambda(t)(x) \cdot x^{(t-1)(n-k)} \cdot (\overline{V}(x))^t \equiv \overline{B}(x) \cdot G(x) \pmod{x^{(n-k)}}.
\]

**Proof:** We start with the “only if” part. Suppose that \( Q^{(0)}(x) \) is such that \( Q(x, y) = Q^{(0)}(x) + Q^*(x, y) \) satisfies (2.4)-(2.5). By Lemma 3.1, there exists a (unique) polynomial \( B(x) \) satisfying (3.1)-(3.2). Define \( \overline{B}(x) \) to be the polynomial of degree \( < \ell(n-k) - \tau \) obtained by reversing the order of coefficients in \( B(x) \), namely

\[
\overline{B}(x) = x^{\ell(n-k)-\tau-1} B(x^{-1}).
\]
Consider the (highest) \( \ell(n-k) \) coefficients of \( x^i \) in both sides of (3.1) for \( i \) in the range \( n-\tau \leq i < n-\tau + \ell(n-k) \). Since \( \deg Q^{(0)}(x) \leq N_0 = n - \tau \), each of those coefficients in \( \sum_{t=1}^{\ell} Q^{(t)}(x)(V(x))^t \) must be equal to its counterpart in the right-hand side of (3.1). If we now reverse the order of coefficients in both sides of (3.1), then the coefficients of \( 1, x, \ldots, x^{\ell(n-k)-1} \) should be identical in the resulting two polynomials. Formally,

\[
x^{n-\tau+\ell(n-k)-1} \cdot \sum_{t=1}^{\ell} Q^{(t)}(x)^t \cdot (V(x)^{-1})^t \equiv x^{\ell(n-k)-1} \cdot B(x^{-1}) \cdot \prod_{j=1}^n (1-\alpha_j x) \pmod{x^{\ell(n-k)}}.
\]

Using the definitions (3.3) and (3.6), the equation (3.7) becomes (3.5).

As for the “if” part, suppose that (3.5) holds for \( \overline{B}(x) \in F_{\ell(n-k)- \tau}[x] \). Define \( B(x) \) to be the polynomial in \( F_{\ell(n-k)- \tau}[x] \) that is obtained by reversing the order of coefficients in \( B(x) \). If we reverse each side of (3.5), we get two polynomials of degree less than \( n - \tau + \ell(n-k) \) that may differ only in their lowest \( n - \tau = N_0 \) coefficients. In other words, there exists some (unique) polynomial \( Q^{(0)}(x) \in F_{N_0}[x] \) such that \( Q^{(0)}(x) + \sum_{t=1}^{\ell} Q^{(t)}(x)(V(x))^t = B(x) \cdot \prod_{j=1}^n (x-\alpha_j) \). The bivariate polynomial \( Q(x,y) = Q^{(0)}(x) + Q^*(x,y) \) thus satisfies (2.4)–(2.5), as required.

For \( t \in [\ell] \), let \( S^{(t)}_\infty(x) = \sum_{i=0}^{\infty} S^{(t)}_i x^i \) be the formal power series which is defined by

\[
\frac{(V(x))^t}{G(x)} = x^{(t-1)(n-1)} \cdot S^{(t)}_\infty(x) + U^{(t)}(x),
\]

where \( U^{(t)}(x) \in F_{\ell(1)(n-1)}[x] \). Indeed, since \( G(0) = 1 \), (3.8) is well-defined (see, for example, [26, Ch. 8]). Further, define the univariate syndrome polynomials \( S^{(t)}(x) = \sum_{i=0}^{n-2-t(k-1)} S^{(t)}_i x^i \) and the bivariate syndrome polynomial \( S(x,y) = \sum_{t=1}^{\ell} S^{(t)}(x,y)^t \).

**Proposition 3.3** Let \( Q^*(x,y) = \sum_{t=1}^{\ell} Q^{(t)}(x)y^t \) satisfy (3.4) and let \( (\Lambda^{(t)}(x))_{t=1}^{\ell} \) be defined by (3.3). There exists a (unique) polynomial \( Q^{(0)}(x) \) such that \( Q(x,y) = Q^{(0)}(x) + Q^*(x,y) \) satisfies (2.4)–(2.5) if and only if there exists a (unique) polynomial \( \Omega(x) \in F_{n-k-\tau}[x] \) that satisfies the EKE

\[
\sum_{t=1}^{\ell} \Lambda^{(t)}(x) \cdot x^{(t-1)(k-1)} \cdot S^{(t)}_\infty(x) \equiv \Omega(x) \pmod{x^{n-k}}.
\]

**Proof:** By Lemma 3.2, all we need to prove is that the EKE (3.9) is equivalent to (3.5). Substituting

\[
(\nabla(x))^t = (x^{(t-1)(n-1)} \cdot S^{(t)}_\infty(x) + U^{(t)}(x)) \cdot G(x)
\]

into (3.5) and rearranging terms yields

\[
\sum_{t=1}^{\ell} \Lambda^{(t)}(x) \cdot x^{(t-1)(n-k) + (t-1)(n-1)} \cdot S^{(t)}_\infty(x) \cdot G(x) \equiv \tilde{V}(x) \cdot G(x) \pmod{x^{\ell(n-k)}},
\]

where

\[
\tilde{V}(x) = \nabla(x) - \sum_{t=1}^{\ell} \Lambda^{(t)}(x) \cdot x^{(t-1)(n-k)} \cdot U^{(t)}(x).
\]
Now,

\[(\ell - t)(n - k) + (t - 1)(n - 1) = (\ell - 1)(n - k) + (t - 1)(k - 1)\]

and

\[\deg \Lambda^{(t)}(x) \cdot x^{(t-\ell)(n-k)} \cdot U^{(t)}(x) = \delta + (\ell - t)(n - k) + (t - 1)(n - 1) - 1 = \ell(n - k) - \tau.\]

Hence, \(\deg \bar{V}(x) < \ell(n - k) - \tau\) if and only if \(\deg \bar{V}(x) < \ell(n - k) - \tau\). Since the polynomials \(G(x)\) and \(x^{(n-k)}\) are relatively prime, we can rewrite (3.10) as

\[
\sum_{t=1}^{l} \Lambda^{(t)}(x) \cdot x^{(t-1)(n-k)+(t-1)(k-1)} \cdot S_{\infty}^{(t)}(x) \equiv \bar{V}(x) \pmod{x^{(n-k)}}. \tag{3.11}
\]

The left-hand side of (3.11) is divisible by \(x^{(t-1)(n-k)}\) and, therefore, so must \(\bar{V}(x)\). Letting \(\Omega(x) = \bar{V}(x)/x^{(t-1)(n-k)}\), we get

\[
\sum_{t=1}^{l} \Lambda^{(t)}(x) \cdot x^{(t-1)(k-1)} \cdot S^{(t)}(x) \equiv \Omega(x) \pmod{x^{(n-k)}}, \tag{3.12}
\]

where we have replaced \(S_{\infty}^{(t)}(x)\) by \(S^{(t)}(x)\), since the coefficients of \(S_{\infty}^{(t)}(x)\) that actually appear in (3.11) are those that correspond to the powers \(x^i\) for \(0 \leq i < n - 1 + t(k - 1)\). Observe that \(\deg \Omega(x) = \deg \bar{V}(x) - (\ell - 1)(n - k)\), so we indeed have \(\deg \Omega(x) < n - k - \tau\) if and only if \(\deg \bar{V}(x) < \ell(n - k) - \tau\).

\[\blacksquare\]

### 3.3 From EKE to homogenous linear equations

In view of the analysis presented in Section 3.2, Step S1 in Sudan’s algorithm splits into two steps, which may be denoted by D0 and D1, similarly to their counterparts in the classical decoding scheme. In Step D0, the bivariate syndrome polynomial \(S(x, y) = \sum_{t=1}^{l} S^{(t)}(x)y^t\) is computed, and in Step D1, the bivariate polynomial \(Q(x, y) = \sum_{t=0}^{l} Q^{(t)}(x)y^t\) is found by solving the EKE. The following proposition shows that the syndrome elements can be computed in Step D0 using a formula which is a generalization of (1.4).

**Proposition 3.4** Let \(\eta_1, \eta_2, \ldots, \eta_n\) be as in (1.5) and \(S_{\infty}^{(t)}(x) = \sum_{i=0}^{\infty} S_i^{(t)} x^i\) as in (3.8). Then

\[
S_i^{(t)} = \sum_{j=1}^{n} v_j^{(t)} \eta_j \alpha_j^i, \quad t \in [\ell], \quad i \geq 0. \tag{3.13}
\]

**Proof:** For each \(t \in [\ell]\), let \(\vec{V}^{(t)}(x) \in F_n[x]\) and \(\bar{U}^{(t)}(x) \in F_{(t-1)(n-1)}[x]\) be the unique polynomials that satisfy

\[
(\vec{V}(x))^t = \bar{U}^{(t)}(x)G(x) + \vec{V}^{(t)}(x), \tag{3.14}
\]

where \(\vec{V}(x)\) is as defined by (3.3). Since \(\vec{V}^{(t)}(\alpha_j^{-1}) = (\vec{V}(\alpha_j^{-1}))^t = (v_j \alpha_j^{-(n-1)})^t\), we can express \(\vec{V}^{(t)}(x)\) as an interpolation polynomial by

\[
\vec{V}^{(t)}(x) = \sum_{j=1}^{n} v_j^{(t)} \alpha_j^{(1-t)(n-1)} \eta_j \prod_{r \in [n] \setminus \{j\}} (1 - \alpha_r x).
\]

32
This, in turn, implies

$$
\frac{\nabla^{(t)}(x)}{G(x)} = \sum_{j=1}^{n} v_j^t \alpha_j^{(1-t)(n-1)} = \sum_{i=(1-t)(n-1)}^{\infty} \sum_{j=1}^{n} v_j^t \eta_j \alpha_j^{i+(t-1)(n-1)}.
$$

(3.15)

Combining (3.8), (3.14), and (3.15) we obtain

$$
x^{(t-1)(n-1)} \cdot S_{\infty}^{(t)}(x) = \frac{(\nabla(x))^t}{G(x)} - U^{(t)}(x) = \frac{\nabla^{(t)}(x)}{G(x)} - U^{(t)}(x) + \tilde{U}^{(t)}(x) = x^{(t-1)(n-1)} \cdot \sum_{i=(1-t)(n-1)}^{\infty} \sum_{j=1}^{n} v_j^t \eta_j \alpha_j^{i} x^i - U^{(t)}(x) + \tilde{U}^{(t)}(x),
$$

that is, $S_{\infty}^{(t)}(x) = \sum_{i=0}^{\infty} S_i^{(t)} x^i = \sum_{i=0}^{\infty} \sum_{j=1}^{n} v_j^t \eta_j \alpha_j^i x^i$.

The time complexity of Step D0, in which all the coefficients of the bivariate syndrome polynomial $S(x, y)$ are computed by (3.13) is at most $\ell$ times the complexity of computing the syndrome in the classical case using (1.4), namely $O(\ell n \log^2 n)$ (see the discussion in Section 1.3).

Writing $\Lambda^{(t)}(x) = \sum_{s=0}^{N_t-1} \Lambda_s^{(t)} x^s$ and $Q^{(t)}(x) = \sum_{s=0}^{N_t-1} Q_s^{(t)} x^s = \sum_{s=0}^{N_t-1} \Lambda_s^{(t)} x^{N_t-1-s}$, from (3.9) we obtain

$$
\sum_{i=1}^{\ell} \sum_{s=0}^{N_t-1} \Lambda_s^{(t)} \cdot S_i^{(t)}(x_{i-1(k-1)i-s}) = 0, \quad n-k-\tau \leq i < n-k,
$$

(compare with (1.6)) or

$$
\sum_{i=1}^{\ell} \sum_{s=0}^{N_t-1} Q_s^{(t)} \cdot S_{i+s}^{(t)} = 0, \quad 0 \leq i < \tau.
$$

(3.16)

(3.17)

For two bivariate polynomials $a(x) = \sum_{s,t} a_{s,t} x^s y^t$ and $b(x) = \sum_{s,t} b_{s,t} x^s y^t$, we define the inner product $\langle a(x, y), b(x, y) \rangle$ by $\sum_{s,t} a_{s,t} b_{s,t}$. Using this notation, (3.17) takes the following short-hand form

$$
\langle x^\rho Q^s(x, y), S(x, y) \rangle = 0, \quad 0 \leq \rho < \tau.
$$

(3.18)

The number of linear equations expressed by (3.18) (or (3.17)) is $\tau$, namely, the maximum number of errors that we attempt to correct, and by (2.9) it is smaller than the number of unknowns in those equations. Comparing (3.18) to the set of linear equations derived directly from (2.4)–(2.5), we conclude that both the number of equations and the number of unknowns have been reduced by $n-\tau$.

Similarly to the classical case, the unknown polynomial coefficients in (3.17) and (3.18) can be considered as defining a multi-level shift register which generates a sequence of zeroes when fed with $\ell$ (syndrome) sequences, as illustrated in Figure 3.1 (compare it to Figure 1.1).
3.4 FIA-based algorithm for solving the EKE

In this section, we derive an algorithm that solves (3.18) for the coefficients of \((Q^{(t)}(x))_{i=1}^{l}\). Equivalently (by reversal), the algorithm solves (3.16) for the coefficients of \((\Lambda^{(t)}(x))_{i=1}^{l}\). Hence, the algorithm presented here is in effect a method for solving the EKE.

Let \(\prec\) denote the (total) order defined in [34] over the set of pairs \(\{(i, t) \mid i \in \mathbb{N}, t \in [\ell]\}\); that is,

\[
(i, t) \prec (i', t') \quad \text{if and only if} \quad \begin{cases} 
  i + t(k-1) < i' + t'(k-1) \\
  (i + t(k-1) = i' + t'(k-1) \quad \text{and} \quad t < t')
\end{cases}
\]

The notation \((i, t) \preceq (i', t')\) means that either \((i, t) = (i', t')\) or \((i, t) \prec (i', t')\), and \(\triangleright\) \((i, t)\) is the pair that immediately follows \((i, t)\) with respect to the order defined by \(\prec\).

We reformulate the linear equations in (3.18) through the \(\tau \times \sum_{t=1}^{\ell} N_t\) matrix \(S\) defined as follows. The columns of \(S\) are indexed by ordered pairs \((\epsilon, \sigma)\), where \(\sigma \in [\ell]\) and \(0 < \epsilon < N_{\sigma}\), and are ordered from left to right with respect to the order \(\prec\) on their indexes. A column in \(S\) indexed by \((\epsilon, \sigma)\) is called a column of type \(\sigma\) and is given by \(S_{(\epsilon, \sigma)} = (S^{(\sigma)}_{\epsilon}, S^{(\sigma)}_{\epsilon+1}, \ldots, S^{(\sigma)}_{\epsilon+\tau-1})^T\). For instance, when \(\ell = 2\) the matrix \(S\) takes the form (1.8).

Denoting by \(Q^{(t)}\) the column vector \((Q^{(0)}(x), Q^{(1)}(x), \ldots, Q^{(\ell)}_{N_{\ell}-1})^T\), and then defining \(Q\) to be the column vector consisting of all the \(N = \sum_{t=1}^{\ell} N_t\) components in \(Q^{(1)}, Q^{(2)}, \ldots, Q^{(\ell)}\), ordered by \(\prec\), we can write (3.17) in a matrix form as

\[
S_{\tau \times N} Q_{N \times 1} = 0_{\tau \times 1},
\]

where the subscripts indicate the dimensions of the respective matrices and vectors. When \(\ell = 1\) (the classical case), the matrix equation (3.19) reduces to (1.7).
We next re-formulate the FIA to fit the particular structure of the matrix $S$. Figure 3.2 presents an algorithm for solving (3.19). If we disregard the starred lines (lines 8-10 and line 19) the resulting algorithm is essentially the FIA of Figure 2.1, tailored for the special form of the matrix $S$ defined above. The vanishing linear combination of $S$ is written in a form of a bivariate polynomial $T(x, y)$ that satisfies $\langle x^\rho T(x, y), S(x, y) \rangle = 0$ for every $0 \leq \rho < \tau$. Lines 3-11 are executed when starting examining a new column, wherein lines 4 and 6 have been shifted forward from their original location in Figure 2.1 through the introduction of the variable compute, which spares the computation of the first discrepancy in any new column. Lines 21-22 compute the index $(\mu, \nu)$ of the next column to be examined.

We describe next how we achieve complexity reduction when the starred lines in Figure 3.2 are incorporated. As was the case with Hankel matrices discussed in Section 2.4, the idea is to start off a new column in $S$ with a clever choice of an initial value for $T(x, y)$. Specifically, suppose we start examining column $(\mu, \nu)$ where $\mu \geq N_1 - N_\nu + 1$; the latter inequality implies that column $(\mu - 1, \nu)$ exists in $S$. Let $\varrho$ be the last value taken by the row pointer $\rho$ when column $(\mu - 1, \nu)$ was scanned; notice that the last value taken by $T(x, y)$ at that column is now written in $A[\varrho](x, y)$. Select $T(x, y) = x \cdot A[\varrho](x, y)$ when starting column $(\mu, \nu)$. Clearly, we have

$$\langle x^{i-1} \cdot T(x, y), S(x, y) \rangle = \langle x^i \cdot A[\varrho](x, y), S(x, y) \rangle$$

for every $i \geq 1$. On the other hand, Lemma 2.6, when applied to the special form of $S$, yields $\langle x^{i-1} \cdot A[\varrho](x, y), S(x, y) \rangle = 0$ for all $i \in [\varrho-1]$. Hence, the initial value of $T(x, y)$ at column $(\mu, \nu)$ already satisfies $\langle x^{i-1} \cdot T(x, y), S(x, y) \rangle = 0$ for $i \in [\varrho-2]$, which means that the discrepancy values will be zero for all $\rho \in [\varrho-2]$. Thus, we can start examining column $(\mu, \nu)$ with $\rho = \varrho - 1$. The book-keeping of the values of $\varrho$ is made in Figure 3.2 through the array $R$.

Figure 3.3 presents a typical sequence of values taken by $(\rho, (\mu, \nu))$ when the algorithm in Figure 3.2 is applied with $\ell = 2$. The solid line corresponds to computations made on columns with indexes of the form $(\mu, 1)$ (namely, columns of $S^{(1)}$), and the dotted line corresponds to columns indexed by $(\mu, 2)$ (columns of $S^{(2)}$). The latter columns start appearing in $S$ only following the dashed vertical line.

**Proposition 3.5** The time complexity of the algorithm in Figure 3.2 is $O(\ell \tau^2)$.

**Proof:** By (2.9), one iteration of the main loop in Figure 3.2 has time complexity $O(\tau)$ (in fact, we never need to scan past the first $\tau + 1$ columns of $S$).

We next bound the number of iterations of the main loop. We say that an iteration is of type $t$ if $\nu = t$ upon starting that iteration. Fix $t \in [\ell]$, and consider the values of $\rho$ and $\mu$ at the end of an iteration of type $t$, compared to their values at the end of the previous iteration of the same type. Those values can change only in one of the following ways:

- the value of $\rho$ may decrease by 1 in line 10, and then —
- (exactly) one of the variables, $\rho$ or $\mu$, is increased by 1 (in line 15 or 22, respectively).

Now, $\mu$ ranges between 0 and $N_\ell - 1$ and its value never decreases. Hence, there are at most $N_\ell$ iterations of type $t$ in which $\mu$ is increased.
Input: Bivariate polynomial \( S(x, y) = \sum_{t=1}^{\ell} S^{(t)}(x) y^t \), where \( S^{(t)}(x) \in F_{r+N_1-1}[x] \);

Data structures:
- Bivariate polynomial \( T(x, y) = \sum_{t=1}^{\ell} T^{(t)}(x) y^t \), where \( T^{(t)}(x) \in F_{N_1}[x] \);
- Variable \( \Delta \in F \);
- Row pointer \( \rho \in [r] \) and column pointer \((\mu, \nu)\);
- Array \( A \) of \( \tau \) bivariate polynomials, each of the same type as \( T(x, y) \);
- Array \( D \) of \( \tau \) entries in \( F \);
- Array \( R \) of \( \ell \) row pointers in \([r]\);
- Variable \( \text{compute} \in \{\text{TRUE}, \text{FALSE}\} \);

Initialize:
- For every \( \rho \in [r] \) do \( D[\rho] \leftarrow 0 \);
- For every \( \nu \in [\ell] \) do \( R[\nu] \leftarrow 0 \);
- \( \rho \leftarrow 1 \); \((\mu, \nu) \leftarrow (0,1)\);
- \( \text{compute} \leftarrow \text{FALSE} \);

Do forever \{ 
1. if \( \text{compute} \) then \( \Delta \leftarrow \langle x^{p-1} \cdot T(x, y), S(x, y) \rangle \);
2. else \{ 
3. \hspace{1em} if \( R[\nu] \leq 1 \) then \{ 
4. \hspace{2em} T(x, y) \leftarrow y^\nu \cdot x^{\mu+N_1-N_1} \;;
5. \hspace{2em} \Delta \leftarrow S(\nu) \cdot x^{\mu+N_1-N_1} \;;
6. \hspace{2em} \rho \leftarrow 1 \;;
7. \hspace{2em} \}
3. \hspace{1em} else \{ 
8. \hspace{2em} T(x, y) \leftarrow x \cdot A[R[\nu]](x, y) \;;
9. \hspace{2em} \Delta \leftarrow D[\rho] \cdot T(x, y) \;;
10. \hspace{2em} \rho \leftarrow R[\nu] - 1 \;;
11. \hspace{2em} \}
12. \hspace{1em} \text{compute} \leftarrow \text{TRUE} \;;
13. \}
14. \hspace{1em} if \( \Delta = 0 \) or \( D[\rho] \neq 0 \) then \{ 
15. \hspace{2em} if \( \Delta \neq 0 \) then \( T(x, y) \leftarrow T(x, y) - \frac{\Delta}{D[\rho]} \cdot A[\rho](x, y) \;;
16. \hspace{2em} if \( \rho = \tau \) then \{ \hspace{1em} \text{return } T(x, y) \;; \text{ exit}; \}
17. \hspace{2em} else \( \rho \leftarrow \rho + 1 \;;
18. \}
19. \hspace{1em} else \hspace{1em} \Delta \neq 0 \) and \( D[\rho] = 0 \) \*/ 
20. \hspace{2em} A[\rho](x, y) \leftarrow T(x, y) \;;
21. \hspace{2em} D[\rho] \leftarrow \Delta \;;
22. \hspace{2em} R[\nu] \leftarrow \rho \;;
23. \hspace{2em} \text{compute} \leftarrow \text{FALSE} \;;
24. \hspace{2em} \nu \leftarrow \nu + 1 \;;
25. \hspace{2em} if \( \nu > \ell \) or \( \mu < N_1 - N_1 \) then \((\mu, \nu) \leftarrow (\mu+1,1)\);
26. \}
\}

Figure 3.2: FIA-based algorithm for solving (3.18).
Figure 3.3: Typical values of \((\rho, (\mu, \nu))\) for \(\ell = 2\) in algorithm of Figure 3.2.

As for the variable \(\rho\), notice that line 10 can be applied only when \(\mu\) was increased in the previous type-\(t\) iteration. Therefore, \(\rho\) can be decreased at most \(N_t\) times. As its initial value is 1 and its final value cannot exceed \(\tau\), it follows that the number of type-\(t\) iterations in which \(\rho\) is increased is at most \(\tau + N_t\). Hence, the number of type-\(t\) iterations is at most \(\tau + 2N_t\).

Summing over \(t\), the number of iterations of the main loop in Figure 3.2 is at most \(\ell\tau + 2\sum_{t=1}^{\ell} N_t = O(\ell\tau)\). The overall time complexity of the algorithm is therefore \(O(\ell\tau^2)\).

The size of the array \(A\) in the algorithm in Figure 3.2 is \(\tau(\tau+1)\), which means that this algorithm has quadratic space complexity.

### 3.5 Improved algorithm for solving the EKE

In this section, we present another algorithm for solving (3.18). The algorithm in Figure 3.4 improves on the FIA-based algorithm in Figure 3.2 in several aspects, and is therefore referred to here as the *improved algorithm*. Instead of computing a single polynomial \(T(x, y)\) which solves (3.18), we compute up to \(\ell\) solution polynomials, \(T_1(x, y), T_2(x, y), \ldots, T_\ell(x, y)\), each of them is minimal by means of leading coefficient, as explained in the sequel. An additional *single* polynomial, denoted \(R(x, y)\), is used in order to keep old values of \(T_\nu(x, y)\). To update the value of \(T_\nu(x, y)\), for any given \(\nu\), the current values of \(T_\nu(x, y)\) and \(R(x, y)\) are used. The polynomial \(R(x, y)\) thus replaces the polynomial array \(A\) from the algorithm in Figure 3.2. The space complexity of the improved algorithm is only \(O(\ell\tau)\) (in comparison with \(O(\tau^2)\) in the FIA-based algorithm).
Roughly speaking, the simplicity of the improved algorithm (compare the number of lines in Figure 3.2 to that in Figure 3.4), as well as the reduced space complexity, are obtained by changing the order of operations in the FIA: $\ell$ different columns (of different types) in $S$ are visited simultaneously by the algorithm according to the order $\prec$ defined on the indexes of the syndrome elements. More details about the connections between the improved algorithm and the FIA are indicated in the proof of Proposition 3.6 below.

For a nonzero bivariate polynomial $T(x, y) = \sum_{t=1}^{\ell} \sum_{i} T_{t}^{(i)} x^{i} y^{t}$, we define $\text{lead}_{y}(T(x, y))$ as the maximal pair $(\mu, \nu)$, with respect to $\prec$ for which $T_{\mu}^{(\nu)} \neq 0$ and denote $\text{lead}_{y}(T(x, y)) = \mu$ and $\text{lead}_{y}(T(x, y)) = \nu$ (lead$(0)$ is defined to be $(-\infty, -\infty)$).

The algorithm in Figure 3.4 scans the syndrome elements $S_{\mu}^{(\nu)}$ in the order defined by $\prec$ on $(\mu, \nu)$, and maintains up to $\ell$ bivariate polynomials $T_{1}(x, y), T_{2}(x, y), \ldots, T_{\ell}(x, y)$, where $\text{lead}_{y}(T_{\nu}(x, y)) = \nu$. An invariant of the algorithm is that $\langle x^{\rho} \cdot T_{\nu}(x, y), S(x, y) \rangle = 0$ for $0 \leq \rho < \rho'$. Lines 6 and 11 update $T_{\nu}(x, y)$ so that it generates the syndrome element $S_{\mu}^{(\nu)}$ as well.

As in Sakata’s algorithm [40], the syndrome elements can be considered as located in a two dimensional array, where row $\nu$ of the array contains the coefficients of $S_{\mu}^{(\nu)}(x)$. Note, however, that in our algorithm, unlike in Sakata’s, $T_{\nu}(x, y)$ is required to generate the syndrome elements in row $\nu$ only, namely $\text{lead}_{y}(T_{\nu}(x, y))$ multiplies elements $S_{\mu}^{(\nu)}$ in the vanishing inner products $\langle x^{\rho} \cdot T_{\nu}(x, y), S(x, y) \rangle$. In Sakata’s algorithm, the polynomial solutions are supposed to ‘move along’ the rows, as well as the columns, of the two-dimensional array and to generate the syndrome elements in the various rows. Hence, our improved algorithm solves a different problem than that solved in [40].

The update of $T_{\nu}(x, y)$ in line 6 does not change the value of $\text{lead}_{y}(T_{\nu}(x, y))$, whereas in line 11, $\text{lead}_{y}(T_{\nu}(x, y))$ grows (with respect to $\prec$), and the value of $T_{\nu}(x, y)$ right before the update is stored in the auxiliary polynomial $R(x, y)$. Unlike Sakata’s algorithm (but similarly to Massey’s algorithm), only one stored polynomial $R(x, y)$ is needed in every stage of our algorithm in order to update any of the polynomials $T_{1}(x, y), T_{2}(x, y), \ldots, T_{\ell}(x, y)$.

Based on Lemma 2.1 and on the analysis in Sections 3.2–3.3, the validity of the algorithm in Figure 3.4 is implied by Proposition 3.6 below. Note that for our purpose of solving (3.18), any one of the polynomials returned by the algorithm will suffice.

**Proposition 3.6** For every $s \in \{1, \ldots, \ell\}$, if there exists some nonzero polynomial $Q^{*}(x, y)$ with $\text{lead}_{y}(Q^{*}(x, y)) = s$ that satisfies (3.4) and (3.18), then the algorithm in Figure 3.4 returns such a polynomial with a minimal value (with respect to $\prec$) of $\text{lead}(Q^{*}(x, y))$.

The proof of Propositions 3.6 makes use of Lemma 3.7 and Lemma 3.8 below. We omit the proof of Lemma 3.7 as it is similar to proofs already contained in [28], [34], and [40]. Throughout the analysis, we use the term iteration of type $s$ to mean an iteration of the algorithm in Figure 3.4 in which the variable $\nu$ takes the value $s$. If in addition $s \in \mathcal{L}$, the iteration will said to be nontrivial.

**Lemma 3.7** Let $S(x, y) = \sum_{t=1}^{\ell} S^{(t)}(x) y^{t}$, $T(x, y) = \sum_{t=1}^{\ell} T^{(t)}(x) y^{t}$, and $R(x, y) = \sum_{t=1}^{\ell} R^{(t)}(x) y^{t}$ be bivariate polynomials, let $\Delta$ and $\delta$ be elements of $F$, and let $\rho$ and $\rho'$ be nonnegative integers such that $\text{lead}(x^{\rho} \cdot R(x, y)) < \text{lead}(x^{\rho} \cdot T(x, y))$ and

$$\langle x^{\rho} \cdot T(x, y), S(x, y) \rangle = 0, \quad 0 \leq \rho < \rho', \quad \text{and} \quad \langle x^{\rho} \cdot T(x, y), S(x, y) \rangle = \Delta,$$
**Input:** Bivariate polynomial \( S(x, y) = \sum_{t=1}^{\ell} S^{(t)}(x) y^t \), where \( S^{(t)}(x) \in F_{\tau+N_t-1}[x] \);

**Data structures:**
- \( \ell \) bivariate polynomials \( T_s(x, y) = \sum_{t=1}^{\ell} T_s^{(t)}(x) y^t \), \( s \in [\ell] \), where \( T_s^{(t)}(x) \in F_{N_s}[x] \);
- index \((\mu, \nu)\) such that \( \nu \in [\ell] \) and \( \mu \in \{0, 1, \ldots, \tau+N_\nu-2\} \);
- variable \( \Delta \in F \);
- integer variables \( \rho, \varrho \);
- bivariate polynomial \( R(x, y) = \sum_{t=1}^{\ell} R^{(t)}(x) y^t \), where \( R^{(t)}(x) \in F_{N_s}[x] \);
- set of indexes \( \mathcal{L} \subseteq [\ell] \);

**Initialize:**
for \( s = 1 \) to \( \ell \) do \( T_s(x, y) \leftarrow y^s \);
\( (\mu, \nu) \leftarrow (0, 1) \);
\( R(x, y) \leftarrow 0 \);
\( \varrho \leftarrow -1 \);
\( \mathcal{L} \leftarrow [\ell] \);

1. **while** \( \mathcal{L} \neq \emptyset \) **do**
   2. if \( \nu \in \mathcal{L} \) then
   3. \( \rho \leftarrow \mu - \text{lead}_x(T_\nu(x, y)) \);
   4. \( \Delta \leftarrow \langle x^\rho \cdot T_\nu(x, y), S(x, y) \rangle \);
   5. if \( \Delta = 0 \) or \( \rho \leq \varrho \) then
   6. if \( \Delta \neq 0 \) then
   7. \( T_\nu(x, y) \leftarrow T_\nu(x, y) - \Delta \cdot x^{\rho-\varrho} \cdot R(x, y) \);
   8. \( \mathcal{L} \leftarrow \mathcal{L} \setminus \{\nu\} \);
   9. **else** /* \( \Delta \neq 0 \) and \( \rho > \varrho \) */
   10. if \( \mu - r < N_\nu \) then
   11. \( \left( \begin{array}{c}
   T_\nu(x, y) \\
   R(x, y)
   \end{array} \right) \leftarrow \left( \begin{array}{c}
   x^{\rho-\varrho} \cdot T_\nu(x, y) - \Delta \cdot R(x, y) \\
   \Delta^{-1} \cdot T_\nu(x, y)
   \end{array} \right) \);
   12. \( \varrho \leftarrow \rho \);
   13. **else** \( \mathcal{L} \leftarrow \mathcal{L} \setminus \{\nu\} \);
   14. \( (\mu, \nu) \leftarrow \prec (\mu, \nu) \);

**Figure 3.4:** Improved algorithm for solving (3.18).
\( \langle x^a \cdot R(x, y), S(x, y) \rangle = 0 \), 0 \( a < \rho \), and \( \langle x^\rho \cdot R(x, y), S(x, y) \rangle = \delta \).

(a) Suppose that \( \rho \leq \rho \) and define \( A(x, y) = T(x, y) - \frac{\Delta}{\delta} \cdot x^{\rho - \rho} \cdot R(x, y) \). Then

1. \( \langle x^a \cdot A(x, y), S(x, y) \rangle = 0 \), 0 \( a \leq \rho \);
2. \( \text{lead}(A(x, y)) = \text{lead}(T(x, y)) \); and —
3. \( \text{lead}(x^\rho \cdot R(x, y)) < \text{lead}(x^{\rho+1} \cdot A(x, y)) \).

(b) Suppose that \( \rho > \rho \) and define \( A(x, y) = x^{\rho - \rho} \cdot T(x, y) - \frac{\Delta}{\delta} \cdot R(x, y) \). Then

1. \( \langle x^a \cdot A(x, y), S(x, y) \rangle = 0 \), 0 \( a \leq \rho \);
2. \( \text{lead}(A(x, y)) = \text{lead}(x^{\rho - \rho} \cdot T(x, y)) \); and —
3. \( \text{lead}(x^\rho \cdot T(x, y)) < \text{lead}(x^{\rho+1} \cdot A(x, y)) \).

Lemma 3.8 The algorithm in Figure 3.4 terminates, and each of its output polynomials, if there exist any, may serve as a polynomial \( Q^*(x, y) \) that satisfies (3.18) under the constraint (3.4).

Proof: By the conditions in lines 7 and 10, the number of nontrivial iterations of type \( s \) throughout an execution of the algorithm is always smaller than \( N_s + \tau \). The set \( L \) thus always becomes empty and the algorithm terminates. Now, if a polynomial \( T_s(x, y) = \sum_{\ell=0}^T T_s^\ell(x) y^\ell \) is returned as output in line 8, then, by Lemma 3.7 and the condition in line 7, it may serve as \( Q^*(x, y) \) in (3.18). By the condition in line 10 we have \( \text{lead}(T_s(x, y)) < (N_s, s) \), and from the definition of the order \( < \) we get that \( \text{deg} T_s^\ell(x) < N_t \) for every \( t \in [T] \). The polynomial \( T_s(x, y) \) thus satisfies the degree constraint (3.4).

Proof of Proposition 3.6: We reformulate the requirements of Proposition 3.6 through the \( \tau \times \sum_{t=1}^T N_t \) matrix \( S \) defined in Section 3.4. We show that if \( S_{i, \ell} \) is the first (leftmost) column of type \( s \) in \( S \) that is linearly dependent on previous columns in \( S \), then a polynomial \( Q^*(x, y) \) with \( \text{lead}(Q^*(x, y)) = (\epsilon, s) \) is returned as output by the algorithm. The respective linear dependency is given by the coefficients of \( Q^*(x, y) \); namely, the coefficient of \( x^\ell y^\ell \) in \( Q^*(x, y) \) multiplies the column \( S_{i, \ell} \) in the linear combination. By Lemma 3.8, it suffices to show that if \( \text{lead}(T_s(x, y)) \) takes in the course of the algorithm a value greater than \( (\epsilon, s) \) (with respect to \( < \)), then the column \( S_{i, \ell} \) is linearly independent of previous columns in \( S \) (this applies also to the case where \( \text{lead}(T_s(x, y)) \) was supposed to take a value which is at least \( (N, s) \), thereby reaching line 13).

By Lemma 3.7, line 11 is the only place in the algorithm where \( \text{lead}(T_{\nu}(x, y)) \) can change. Therefore, all we have to show is that whenever line 11 is reached with given values of \( \nu, \rho, \rho \), and \( \text{lead}(T_{\nu}(x, y)) = (\xi, \nu) \), then each of the columns \( S_{\rho, \nu} \), \( \xi \leq p < \xi + \rho - \rho \), is linearly independent of the columns standing to its left in \( S \).

Let \( s_i, \rho_i, \rho_i, \Delta_i, (\xi_i, s_i) \), and \( \{T_{s, i}(x, y)\}_{s=1}^T \) denote the values of \( \nu, \rho, \rho_i, \Delta, \text{lead}(T_{\nu}(x, y)) \), and \( \{T_s(x, y)\}_{s=1}^T \) right before the \( i \)th execution of line 11; note that \( \rho_i = \rho_i - \rho_i \). Let \( Y_i \) be the upper-left sub-matrix of \( S \) that consists of the columns \( S_{\rho, \nu} \) for \( (0, 1) \leq \rho \leq \rho_i \), shortened to their first \( \rho_i + 1 \) entries. Define the sets \( X_i \) inductively as follows:

\[ X_0 = \emptyset \quad \text{and} \quad X_i = X_{i-1} \cup \{(\xi_i + j, s_i) : 0 \leq j < \rho_i - \rho_i \} ; \]

40
observing that \(|X_i| = |X_{i-1}| + \rho_i - \rho_{i-1}\) and that \(\rho_0 \equiv \rho_1 = -1\), we have \(|X_i| = \rho_i + 1\). We denote by \(Z_i\) the \((\rho_i + 1) \times (\rho_i + 1)\) sub-matrix of \(Y_i\) consisting of the columns of the latter indexed by \(X_i\). It can be verified that \(Z_i\) consists of all the columns of \(Y_i\), except, possibly, a certain number of rightmost columns of each type other than \(s_i\) in \(Y_i\).

The rest of the proof is devoted to showing that every column in \(Z_i\) is linearly independent of columns that stand to its left in \(Y_i\). We show this in two steps:

- We first show that each column of \(Y_i\) that is not a column of \(Z_i\) is linearly dependent on previous columns in \(Y_i\).
- We then prove that \(\text{rank}(Y_i) = \rho_i + 1\).

(We point out that in the classical case of \(\ell = 1\), the matrices \(Y_i\) and \(Z_i\) coincide, thereby making the first step vacuous.)

Let \((\epsilon, \sigma)\) be the index of a column in \(Y_i\) that does not belong to \(Z_i\). It is clear that \(\sigma \neq s_i\). Since \((\epsilon, \sigma) \notin X_i\), it follows that \(\text{lead}(T_{\sigma,i}(x, y)) \leq (\epsilon, \sigma)\). Let \(a\) be an integer such that \(0 \leq a \leq \rho_i\). We distinguish between two cases:

**Case 1:** \((\epsilon + a, \sigma) \prec (\xi_0 + \rho_i, s_i)\). Here, the index \((\mu, \nu)\) reaches the value \((\epsilon + a, \sigma)\) with a polynomial \(T_{\sigma,i}(x, y)\) with \(\text{lead}(T_{\sigma,i}(x, y)) = \text{lead}(T_{\sigma,i}(x, y)) \equiv (h, \sigma)\) before \((\mu, \nu)\) takes the value \((\xi_0 + \rho_i, s_i)\) with the polynomial \(T_{\nu}(x, y) = T_{s,i}(x, y)\). In this case we have

\[
\langle x^{\alpha} \cdot x^{e-h} \cdot T_{\sigma,i}(x, y), S(x, y) \rangle = 0 .
\]

**Case 2:** \((\xi_0 + \rho_i, s_i) \prec (\epsilon + a, \sigma)\). To the smallest \(a'\) such that

\[
\langle x^{\alpha'} \cdot x^{e-h} \cdot T_{\sigma,i}(x, y), S(x, y) \rangle \neq 0 ,
\]

we can apply Lemma 3.7(a) with \(T(x, y) \leftarrow x^{e-h} \cdot T_{\sigma,i}(x, y), \rho \leftarrow a', R(x, y) \leftarrow T_{s,i}(x, y), \) and \(\rho \leftarrow \rho_i\), to obtain a polynomial \(A(x, y)\) with \(\text{lead}(A(x, y)) = (\epsilon, \sigma)\) for which

\[
\langle x^{b} \cdot A(x, y), S(x, y) \rangle = 0 , \quad 0 \leq b \leq a' .
\]

By repeatedly applying Lemma 3.7 as in Case 2, we can update the polynomial \(A(x, y)\) while keeping \(\text{lead}(A(x, y)) = (\epsilon, \sigma)\) so that it satisfies

\[
\langle x^{a} \cdot A(x, y), S(x, y) \rangle = 0 , \quad 0 \leq a \leq \rho_i .
\]

The last equation means that the column \((Y_i)_{(\epsilon, \sigma)}\) is linearly dependent on columns standing to its left in \(Y_i\). This completes the first step of our proof.

In our second step, we show that \(\text{rank}(Y_i) = \rho_i + 1\) by applying Gaussian elimination to the columns of \(Y_i\), where the linear combinations applied to the columns will be determined by \(T_{s,i}(x, y)\). By Lemma 3.7 it follows that the sequence of updates carried out on \(T_{\nu}(x, y)\) in the algorithm to produce \(T_{s,i}(x, y)\) guarantees that

\[
\langle x^{a} \cdot T_{s,i}(x, y), S(x, y) \rangle = \begin{cases} 
0 & \text{if } 0 \leq a < \rho_i \\
\Delta_i & \text{if } a = \rho_i
\end{cases} .
\]

This implies that when using the coefficients of \(T_{s,i}(x, y)\) to compute a linear combination of the columns \((Y_i)_P, (0, 1) \leq P \preceq (\xi_0, s_i)\), we end up with a column vector \((0, 0, \ldots, 0, \Delta_i)^T \in \mathbb{F}_{\rho_i + 1}^*\).
Next, we take advantage of the Hankel-like structure of \( S \) and generalize our previous argument as follows. For every \( j \) in the range \( 0 \leq j < \rho_i - \rho_{i-1} \), the linear combination of \( (Y_i)_P, (0,1) \leq P \leq (s_i + j, s_i) \), yields a column vector \((0,0,\ldots,0,\Delta_i,\ldots)^T \in F^{n+1} \), where the number of leading zeros is \( \rho_i - j \). We thus have,

\[
\text{rank}(Y_i) \geq \text{rank}(Y_{i-1}) + \rho_i - \rho_{i-1},
\]

which readily implies by induction the desired result.

**Proposition 3.9** The time complexity of the algorithm in Figure 3.4 is \( O(\ell \tau^2) \) and its space complexity is \( O(\ell \tau) \).

**Proof:** In every nontrivial iteration of the algorithm of Figure 3.4, the most time-consuming steps are the computations of \( \Delta \) in line 4 and the polynomial updates in lines 6 and 11. The time complexity of all those computations is linear in the number of nonzero coefficients of the respective polynomial \( T_s(x,y) \). The check in line 10 guarantees that \( \text{lead}(T_s(x,y)) \prec (N_s, s) \) and, so, the number of nonzero coefficients in \( T_s(x,y) \) never exceeds \( \sum_{t=1}^{\ell_0} N_t \). By (2.9), the number of coefficients in \( T_s(x,y) \), as well as the time complexity of every computation in lines 4, 6, and 11, is \( O(\tau) \).

As shown in the proof of Lemma 3.8, the number of nontrivial iterations of type \( s \) is smaller than \( N_s + \tau \) for every \( s \in [\ell] \). The overall number of nontrivial iterations of all types throughout the execution of the algorithm can therefore be bounded from above by \( \ell \tau + 2 \sum_{t=1}^{\ell_0} N_t = O(\ell \tau) \), where the equality follows from (2.9). The time complexity of the whole algorithm is thus \( O(\ell \tau^2) \).

As for the space complexity, most of the memory is allocated for the \( \ell + 1 \) polynomials \( (T_s(x,y))_{s=1}^{\ell_0} \) and \( R(x,y) \), where for each of them we allocate \( \sum_{t=1}^{\ell_0} N_t = O(\tau) \) coefficients over \( F \).

### 3.6 Computing \( Q^{(0)}(x) \)

To complete Step D1, we need to compute the polynomial \( Q^{(0)}(x) \) for which \( Q^{(0)}(x) + Q^*(x,y) = Q(x,y) \) satisfies (2.4)–(2.5) (see Proposition 3.3). Since

\[
Q^{(0)}(\alpha_j) = -Q^*(\alpha_j,v_j) = -\sum_{t=1}^{\ell} Q^{(t)}(\alpha_j)v_j^t, \quad j \in [n],
\]

the polynomial \( Q^{(0)}(x) \) can be obtained by interpolation once we compute the right-hand side of (3.20) for \( N_0 = n - \tau \) pairs \((\alpha_j,v_j)\).

As for the time complexity of this computation, note that \( \deg Q^{(t)}(x) \leq \deg Q^{(0)}(x) < N_0 \) for every \( t \in [\ell] \) and that by (2.8)

\[
N_0 \leq \frac{2(n + \ell + 1)}{\ell + 1}.
\]

Hence, for every \( t \in [\ell] \), we can evaluate \( Q^{(t)}(x) \) at \( n - \tau \) locators \( \alpha_j \) in time complexity \( O((n/\ell) \log^2 n) \). So, the right-hand side of (3.20) can be evaluated for \( n - \tau \) pairs \((\alpha_j,v_j)\) in time \( O(n \log^2 n) \); this will also be the time complexity of interpolating \( Q^{(0)}(x) \) out of those computed values.
Chapter 4

Finding linear factors of bivariate polynomials

4.1 Introduction

In this chapter, we present an efficient implementation of Step S2 in Sudan’s algorithm which applies Lemma 2.2 and looks for all the polynomials \( g(x) \in F_k[x] \) such that \( Q(x, g(x)) \) is identically zero. Specifically, our goal is to compute all the consistent polynomials \( f(x) \in F_k[x] \) for which the vector \( (f(a_1), f(a_2), \ldots, f(a_n)) \) is at Hamming distance \( \leq \tau \) from the received word \( v \).

Throughout the chapter we use the following terminology: The \( y \)-degree of \( Q(x, y) \) is the degree of \( Q(x, y) \) as a polynomial in \( y \) over \( F[x] \). A \( y \)-root of \( Q(x, y) \) over the polynomial ring \( F[x] \) is any polynomial \( g(x) \in F[x] \) for which \( Q(x, g(x)) \) is identically zero.

The recursive procedure \textit{Reconstruct} presented in this chapter finds all the \( y \)-roots in \( F_k[x] \) of any given bivariate polynomial over the finite field \( F \), where \( k \) is some fixed integer. However, \textit{Reconstruct} can be used to solve the more general problem of finding all the \( y \)-roots of a given bivariate polynomial \( Q(x, y) \). This is implied by observing that the degree of any polynomial \( g(x) \) which is a \( y \)-root of \( Q(x, y) = \sum_{t=0}^{l} Q^{(t)}(x)y^t \) can be bounded from above by expressions that depend on the degrees of \( Q^{(t)}(x) \), \( t = 0, 1, \ldots, \ell \). As indicated by M. Kaminski, the \( y \)-root \( g(x) \) divides the polynomial \( Q^{(0)}(x) \), so \( \deg g(x) \leq \deg Q^{(0)}(x) \).

In addition, if \( k \) is the largest integer such that \( \deg Q^{(t)}(x) = \deg Q^{(t)}(x) \geq (\ell-t)(k-1) \), for every \( t < \ell \), then \( \deg g(x) < k \).

4.2 The algorithm

The recursive procedure \textit{Reconstruct} in Figure 4.1 computes a set of up to \( \ell \) polynomials in \( F_k[x] \) that contains as a subset all the \( y \)-roots of \( Q(x, y) \) in \( F_k[x] \); as such, this set also contains all the consistent polynomials. The procedure \textit{Reconstruct} is initially called with parameters \( (Q, k, 0) \), where \( Q = Q(x, y) = \sum_{t=0}^{l} Q^{(t)}(x)y^t \) is a nonzero bivariate polynomial with \( y \)-degree \( \leq \ell \), e.g., a polynomial that satisfies (2.4). The validity of \textit{Reconstruct}, as established in Proposition 4.2 below, is based on the following lemma, which shows that the coefficients of a \( y \)-root \( g(x) \) of \( Q(x, y) \) can all be calculated recursively as roots of univariate
polynomials.

**Procedure** `Reconstruct` (bivariate polynomial $Q(x,y)$, integer $k$, integer $i$)
/* A global array $\phi[0, \ldots, k-1]$ is assumed.
The initial call needs to be with $Q(x, y) \neq 0$, $k > 0$, and $i = 0$.
*/
R1 find the largest integer $e$ for which $Q(x, y)/x^e$ is still a (bivariate) polynomial;
R2 $M(x, y) \leftarrow Q(x, y)/x^e$;
R3 find all the roots in $F$ of the univariate polynomial $M(0, y)$;
R4 for each of the distinct roots $\gamma$ of $M(0, y)$ do
  { 
    R5 $\phi[i] \leftarrow \gamma$;
    R6 if $i = k-1$ then output $\phi[0], \ldots, \phi[k-1]$;
    else
      { 
        R7 $\widehat{M}(x, y) \leftarrow M(x, y + \gamma)$;
        R8 $\widehat{M}(x, y) \leftarrow \widehat{M}(x, xy)$;
        R9 `Reconstruct($\widehat{M}(x, y)$, $k$, $i+1$)``
      }
  }

Figure 4.1: Recursive procedure for finding a superset of the consistent polynomials.

**Lemma 4.1** Let $g(x) = \sum_{s \geq 0} g_s x^s$ be a $y$-root of a nonzero bivariate polynomial $Q(x, y)$ over $F$. For $i \geq 0$, let $\psi_i(x) = \sum_{s \geq i} g_s x^{s-i}$ and let $Q_i(x, y)$ and $M_i(x, y)$ be defined inductively by $Q_0(x, y) = Q(x, y)$,
$$M_i(x, y) = x^{-e_i} Q_i(x, y) \quad \text{and} \quad Q_{i+1}(x, y) = M_i(x, xy + g_i), \quad i \geq 0,$$
where $e_i$ is the largest integer such that $x^{e_i}$ divides $Q_i(x, y)$. Then, for every $i \geq 0$,
$$Q_i(x, \psi_i(x)) = 0 \quad \text{and} \quad M_i(0, g_i) = 0,$$
while $M_i(0, y) \neq 0$.

**Proof:** First observe that the $y$-degrees of the polynomials $Q_i(x, y)$ are the same for all $i$ and, so, $Q_i(x, y) \neq 0$ and $e_i$ is well-defined. Also, since $x$ does not divide $M_i(x, y)$ then $M_i(0, y) \neq 0$. Next, we prove that $Q_i(x, \psi_i(x)) = 0$ by induction on $i$, where the induction base $i = 0$ is obvious. As for the induction step, if $\psi_i(x)$ is a $y$-root of $Q_i(x, y)$, then $\psi_{i+1}(x) = (\psi_i(x) - g_i)/x$ is a $y$-root of $Q_i(x, xy + g_i)$ and hence of $Q_{i+1}(x, y) = M_i(x, xy + g_i) = x^{-e_i} Q_i(x, xy + g_i)$. Finally, by substituting $x = 0$ in $M_i(x, \psi_i(x)) = x^{-e_i} Q_i(x, \psi_i(x)) = 0$ we obtain $M_i(0, g_i) = M_i(0, \psi_i(0)) = 0$.

**Proposition 4.2** Let $Q(x, y)$ be a nonzero bivariate polynomial. Every $y$-root in $F_k[x]$ of $Q(x, y)$ is found by the call `Reconstruct($Q$, $k$, $0$)`.

**Proof:** Using the notations of Lemma 4.1, there is a recursion descend in `Reconstruct` where recursion level $i$ is called with the parameters $(Q_i, k, i)$ and $\phi[i]$ is set to $g_i$. 

44
We show in Proposition 4.5 below that \emph{Reconstruct} outputs at most \( \ell \) polynomials. However, some of those polynomials may not be \( y \)-roots of \( Q(x, y) \). To complete Step 2 in Sudan’s decoding algorithm, each of those polynomials is evaluated at the code locators, producing up to \( \ell \) different codewords from which we select the consistent codewords as those that are at Hamming distance \( \leq \tau \) from the received word \( \mathbf{v} \).

In order to solve the more general problem of providing all the \( y \)-roots in \( F_k[x] \), we compute the coefficients of the univariate polynomial \( Q(x, g(x)) \), for every polynomial \( g(x) \) computed by \emph{Reconstruct}. If these coefficients are all zero, then \( g(x) \) is declared a \( y \)-root.

The computation of \( \tilde{Q}(x) = Q(x, g(x)) \) can be done using Horner’s rule (see [2, chapter 12]), namely

\[
\begin{align*}
\tilde{Q}_0(x) &= Q^{(\ell)}(x) \\
\tilde{Q}_1(x) &= \tilde{Q}_{i-1}(x) \cdot g(x) + Q^{(\ell-i)}(x), \quad i \in [\ell].
\end{align*}
\]

### 4.3 Complexity analysis

This section is devoted to proving Proposition 4.7 below in which the time and space complexities of the recursive procedure \emph{Reconstruct} in Figure 4.1 are established. In each of the recursion levels \( i \) of \emph{Reconstruct}, we find roots of nonzero polynomials \( M(0, y) = M_i(0, y) \) with degree at most \( \ell \). It may seem at first that the number of root extractions could grow exponentially. However, Lemma 4.3 below shows that having more than one root of \( M_i(0, y) \) in a given recursion level is compensated by having a multiple root in the respective polynomial \( M_{i-1}(0, y) \) in the previous recursion level.

**Lemma 4.3** Let \( M_1(x, y) = \sum_{t=0}^{\ell} M^{(t)}(x)y^t \) be a nonzero bivariate polynomial over \( F \) and let \( \gamma \in F \) be a \( y \)-root of multiplicity \( h \) of \( M_1(0, y) \). Define \( M_2(x, y) = x^{-e}M_1(x, xy + \gamma) \), where \( e \) is the largest integer for which \( x^e | M_1(x, xy + \gamma) \). Then \( \deg M_2(0, y) \leq h \).

**Proof:** Similarly to the notations in Figure 4.1, we denote

\[
\tilde{M}(x, y) = \sum_{t=0}^{\ell} \tilde{M}^{(t)}(x)y^t = M_1(x, y + \gamma)
\]

and

\[
\tilde{M}(x, y) = \sum_{t=0}^{\ell} \tilde{M}^{(t)}(x)y^t = M_1(x, xy + \gamma).
\]

Since \( \gamma \) is a root of multiplicity \( h \) of \( M_1(0, y) \), then \( y = 0 \) is a root of multiplicity \( h \) of \( \tilde{M}(0, y) \). Thus, \( \tilde{M}^{(t)}(0) = 0 \) for \( 0 \leq t < h \) and \( \tilde{M}^{(h)}(0) \neq 0 \); equivalently, \( x \) divides \( \tilde{M}^{(t)}(x) \) for \( 0 \leq t < h \) but it does not divide \( \tilde{M}^{(h)}(x) \). Noting that \( \tilde{M}^{(t)}(x) = \tilde{M}^{(t)}(x)x^t \) it follows that \( x \) divides \( \tilde{M}(x, y) \) but \( x^{h+1} \) does not.

The largest integer \( e \) such that \( x^e \) divides \( \tilde{M}(x, y) \) thus satisfies \( 1 \leq e \leq h \). Now,

\[
M_2(x, y) = \frac{\tilde{M}(x, y)}{x^e} = \sum_{t=0}^{e} \frac{\tilde{M}^{(t)}(x)x^t}{x^e} y^t + \sum_{t=e+1}^{\ell} \tilde{M}^{(t)}(x)x^{t-e}y^t.
\]
Substituting $x=0$ in $M_2(x, y)$ yields a univariate polynomial $M_2(0, y)$ of degree $e \leq h$.

**Corollary 4.4** Consider the special case where the very first execution of Step R2 in Reconstruct results in a polynomial $M(x, y)$ such that all the roots in $F$ of $M(0, y)$ are simple. Then the polynomials obtained in Step R3 throughout all the subsequent recursive calls have degree at most 1, meaning that their roots can be found simply by solving linear equations over $F$.

As for the general case, we have the following upper bounds.

**Proposition 4.5** Suppose that Reconstruct is initially called with the parameters $(Q, k, 0)$, where $Q = Q(x, y) = \sum_{t=0}^{r} Q^{(t)}(x)y^t$ is a nonzero bivariate polynomial. Then the number of output polynomials produced by Reconstruct is at most $\ell$ and the overall number of recursive calls made to Reconstruct (in Step R9) is at most $\ell(k-1)$.

**Proof:** For $0 \leq s < k$, denote by $\omega_s$ the sum of the degrees of all the polynomials $M(0, y)$ that Step R3 is applied to when $i$ equals $s$. When $s = 0$, the degree of $M(0, y) = M_0(0, y)$ is at most $\ell$, and by Lemma 4.3 we have $\omega_s \leq \omega_{s-1}$ for every $s \in [k-1]$. It can therefore be proved by induction on $s$ that $\omega_s \leq \ell$. As a result, Reconstruct generates at most $\omega_{k-1} \leq \ell$ outputs, and the number of executions of Step R9 is $\sum_{s=0}^{k-2} \omega_s \leq \ell(k-1)$.

**Lemma 4.6** Assume a call to Reconstruct as in Proposition 4.5, and further assume that $Q(x, y)$ satisfies (2.4). Then the $y$-degree of each of the bivariate polynomials computed in any of the recursion levels is at most $\ell$, and its $x$-degree is at most $m + \ell(k-1) = O(n/\ell)$.

**Proof:** It is easy to see that Steps R2, R7, and R8 never increase the $y$-degree. As for the $x$-degree, let $M_i(x, y)$ be a polynomial computed in Step R2 in recursion level $i$ and write $M_i(x) = \sum_{t=0}^{r} M_i^{(t)}(x)y^t$. The degree of $M_i^{(t)}(x)$ can increase with $i$ only as a result of Step R8, and it is easy to check by induction on $i$ that

$$\deg M_i^{(t)}(x) < N_t + ti = N_0 - t(k-1-i)$$

for every $0 \leq i < k$. So, $\deg M_i^{(t)}(x) < N_0 \leq 2(n + \ell + 1)/\ell$ (see (3.21)).

**Proposition 4.7** Assume a call to Reconstruct as in Proposition 4.5, where $Q(x, y)$ satisfies (2.4). The time complexity of such an application to its full recursion depth is $O((\ell\log^2 \ell)k(n + \ell\log q))$, and it can be implemented using space of overall size $O(n)$.

**Proof:** Each execution of Step R1, R2, or R8 has time complexity which is proportional to the number of coefficients in the polynomials involved in that step. By Lemma 4.6, this number is $O(n)$. By Proposition 4.5, each of those steps is executed at most $\ell k$ times throughout the recursion levels. Therefore, the contribution of Steps R1, R2, and R8 to the overall complexity of Reconstruct is $O(\ell kn)$.

By Proposition 4.5, the sum of the degrees of all the polynomials $M(0, y)$ that Step R3 is applied to at the $i$th recursion level is at most $\ell$. The roots in $F = GF(q)$ of a polynomial of degree $u$ can be found in expected time complexity $O((u^2\log^2 u)\log q)$ [26, Ch. 4],[33];
also recall that there are known efficient deterministic algorithms for root extraction when the characteristic of $F$ is small [4, Ch. 10], and root extraction is particularly simple when $\ell = 2$ [27, pp. 277–278]. Therefore, for any $0 \leq i < k$, the executions of Step R3 at recursion level $i$ have accumulated time complexity $O((\ell^2 \log^2 \ell) \log q)$, and the contribution of Step R3 to the overall complexity of $\text{Reconstruct}$ is $O((\ell^2 \log^2 \ell) k \log q)$.

Let $M(x, y)$ be the polynomial that Step R7 is applied to at some iteration level $i$. Writing $M(x, y)$ as a polynomial in $x$, we get

$$M(x, y) = \sum_{s=0}^{N_0-1} M^{[s]}(y)x^s.$$  

If $\ell_s$ is the smallest integer $t$ such that $N_t + ti \leq s$, then it is easy to check that $\deg M^{[s]}(y) < \ell_s$ for every $0 \leq s < N_0$. The computed polynomial $\widehat{M}(x, y)$ in Step R7 can be written as

$$\widehat{M}(x, y) = \sum_{s=0}^{N_0-1} \widehat{M}^{[s]}(y)x^s = M(x, y + \gamma) = \sum_{s=0}^{N_0-1} M^{[s]}(y + \gamma)x^s,$$

and, so, we can compute each polynomial $\widehat{M}^{[s]}(y)$ by interpolating the values $M^{[s]}(\alpha + \gamma)$ at $\ell_s$ distinct points $\alpha \in F$. Therefore, each polynomial $\widehat{M}^{[s]}(y)$ can be found in time complexity $O(\ell_s \log^2 \ell_s)$. We now observe that $\ell_s \leq \ell$ and that

$$\sum_{s=0}^{N_0-1} \ell_s = \sum_{t=0}^{\ell} (N_t + ti) \leq (\ell+1)N_0 \leq 2(n + \ell + 1),$$

the latter inequality following from (3.21). Hence, each execution of Step R7 has time complexity $O(n \log^2 \ell)$, and the contribution of Step R7 to the overall complexity of $\text{Reconstruct}$ is therefore $O(kn\ell \log^2 \ell)$.

Summing up the contributions of the steps of $\text{Reconstruct}$, the time complexity of an application of $\text{Reconstruct}$ to its full recursion depth is $O((\ell \log^2 \ell) k (n + \ell \log q))$. As for the space complexity, notice that the input parameter $Q(x, y)$ can be recomputed from $e, \gamma$, and the parameter $\widehat{M}(x, y)$ to the next recursion level. So, instead of keeping the polynomials in each recursion level, we can recompute them after each execution of Step R9. Therefore, $\text{Reconstruct}$ can be implemented using space of overall size $O(n)$.

Finally, the time complexity of computing the consistent codewords out of the output polynomials of $\text{Reconstruct}$ is $O(\ell n \log^2 n)$, as the re-encoding involves the evaluation of those polynomials at the code locators. This is also the time complexity needed for finding all the linear factors (and not just the consistent ones), assuming that the polynomial multiplications in (4.1) are carried out using the Discrete Fourier Transform [2, chapter 7], namely through evaluations and interpolations.

The algorithm presented in this chapter applies to general bivariate polynomials, and it can therefore be used to implement Step S2 in Guruswami-Sudan’s algorithm as well. Since the number of coefficients in their algorithm is bounded from above by $r^2n$ (see Section 2.2), and not by $n$, the time complexity of the $\text{Reconstruct}$ procedure becomes

$$O((\ell \log^2 \ell) k (r^2n + \ell \log q)).$$

Assuming $r^2n = O(\ell^2k)$ (see the comment right after Theorem 1.1), the time complexity is

$$O((\ell^2 \log^2 \ell) k (\ell k + \log q)).$$
Chapter 5

Efficient list-decoding algorithm—summary and outline for extension

5.1 Summarizing efficient implementation of Sudan’s algorithm

Figure 5.1 below summarizes the list-decoding algorithm obtained by combining the efficient procedures for interpolation and factorization presented in Chapters 3–4. The three main decoding steps in our algorithm, as it appears in Figure 5.1, are denoted D0, D1, and D2, to point out their relationship with the classical decoding algorithms as outlined in Section 1.3.1. Steps D0 and D1 replace Step S1 in Sudan’s algorithm that finds the bivariate polynomial \( Q(x, y) \).

As shown in Chapter 3, the time complexity of Step D0 is \( O(\ell n \log^2 n) \) and the time complexity of Step D1 is \( O(\ell^2) \). Compared to classical decoding, those figures are larger by a factor of \( \ell \). Step D2, which is presented in Chapter 4, is an efficient application of Step S2 in Sudan’s algorithm and has time complexity \( O((\ell \log^2 \ell) k (n + \ell \log q)) \).

In cases where the particular use of the decoding algorithm does not dictate an upper bound on \( \ell \), we can select the value of \( \ell \) that maximizes (2.1) subject to (2.2). By (2.7) we will thus have \( \ell = O(\sqrt{n/k}) \) (and see also the comment right after Theorem 1.1). For this value of \( \ell \), the time complexities of Steps D0, D1, and D2 are \( O(n^{3/2}k^{-1/2} \log^2 n) \), \( O((n-k)^2 \sqrt{n/k}) \), and \( O((\sqrt{n}k + \log q)n \log^2 (n/k)) \), respectively.

The next example is provided to illustrate the various decoding steps; the parameters were selected to be small enough so that the computation can be more easily verified by the reader.

**Example 5.1** Let \( F = GF(19) \), \( n = 18 \), and \( k = 2 \). When maximizing (2.1) we get \( \tau = 12 \) for \( \ell = 4 \) and \( m = 1 \). We select \( \alpha_j = j \) and obtain from (1.5) that \( \eta_j = -\alpha_j \).

Suppose we encode the polynomial \( f(x) = 18 + 14x \) by (1.3) and get the following transmitted codeword, error vector, and received word:

- \( c = (13, 8, 3, 17, 12, 7, 2, 16, 11, 6, 1, 15, 10, 5, 0, 14, 9, 4) \)
- \( e = (11, 16, 17, 12, 17, 0, 0, 2, 14, 0, 0, 0, 3, 0, 14, 8, 11, 15) \)
- \( v = (5, 5, 1, 10, 10, 7, 2, 18, 6, 6, 1, 15, 13, 5, 14, 3, 1, 0) \)

49
Technion - Computer Science Department - Ph.D. Thesis - PHD-2001-01 - 2001

**Preliminary Step:**
Given $n$, $k$, and distinct code locators $\alpha_1, \alpha_2, \ldots, \alpha_n$ in $F$, fix an upper bound on the allowed number $\ell$ of consistent codewords. Compute $\ell$, $m$, and $\tau$ so that (2.1) is maximized, subject to (2.2) and the upper bound on $\ell$. Compute the multipliers $\eta_i, \eta_2, \ldots, \eta_m$ by (1.5). Define $N_i$ by (2.3).

**Input:** $v = (v_1, v_2, \ldots, v_n)$.

**Step D0:**
Compute the syndrome elements

$$S^{(\ell)}_i = \sum_{j=1}^n v_j^t \eta_i \alpha_j^t, \quad t \in [\ell], \quad 0 \leq i < n + t(k-1) - 1.$$  

**Step D1:**
- Using the algorithm in Figure 3.4, compute a polynomial $Q^*(x, y) = \sum_{t=1}^\ell Q^{(t)}(x)y^t$ that solves the equations

$$\langle x^\rho Q^*(x, y), S(x, y) \rangle = 0, \quad 0 \leq \rho < \tau,$$

under the degree constraints

$$\text{deg} Q^{(t)}(x) < N_i, \quad t \in [\ell].$$

- Compute the polynomial $Q^{(0)}(x) \in F_{N_0}[x]$ by (3.20).

**Step D2:**
- Call *Reconstruct* with the initial parameters $Q(x, y) = Q^{(0)}(x) + Q^*(x, y)$, $k$, and 0.
- For each $g(x) \in F_k[x]$ in the output of *Reconstruct*, compute the corresponding codeword $c = (g(\alpha_1), g(\alpha_2), \ldots, g(\alpha_n))$. Output $c$ if the Hamming distance between $c$ and $v$ is $\tau$ or less.

**Figure 5.1:** Summary of efficient list decoding algorithm.

The computation of the syndrome elements in Step D0 results in the following coefficients of the polynomials $(S^{(0)}(x) = \sum_{t=0}^{r-1} S^{(t)}(x^t))_{t=1}^\ell$:

- $S^{(1)}(x) : (13, 14, 5, 11, 3, 4, 10, 14, 13, 14, 11, 14, 17, 4, 0, 2)$
- $S^{(2)}(x) : (4, 8, 14, 18, 9, 18, 5, 13, 11, 6, 8, 8, 16, 0, 12)$
- $S^{(3)}(x) : (3, 12, 5, 7, 10, 18, 4, 14, 0, 14, 18, 11, 16, 3)$
- $S^{(4)}(x) : (14, 13, 0, 13, 10, 1, 9, 3, 7, 8, 11, 0, 7)$

Step D1 yields the following polynomials $Q^{(t)}(x)$,

- $Q^{(0)}(x) = 4 + 12x + 5x^2 + 11x^3 + 8x^4 + 13x^5$
- $Q^{(1)}(x) = 14 + 14x + 9x^2 + 16x^3 + 8x^4$
- $Q^{(2)}(x) = 14 + 13x + x^2$
- $Q^{(3)}(x) = 2 + 11x + x^2$
- $Q^{(4)}(x) = 17$

where $Q^*(x, y) = \sum_{t=1}^4 Q^{(t)}(x)y^t$ is the first polynomial returned by the algorithm in Figure 3.4 and $Q^{(0)}(x)$ is computed by (3.20).
When applying $Reconstruct$ in Step D/2 to $Q(x, y) = \sum_{t=0}^{4} Q(t)(x)y^t$, we obtain the following four different solutions $g(x)$ for $f(x)$:

$$18 + 14x, \ 18 + 15x, \ 14 + 16x, \ 8 + 8x$$

The first two solutions share the same constant coefficient, 18, which is a multiple root of the polynomial $M(0, y) = Q(0, y) = 4 + 14y + 14y^2 + 2y^3 + 17y^4$ at recursion level $i = 0$. The first solution for $f(x)$ corresponds to the correct codeword. The second solution is not even a $y$-root of $Q(x, y)$ (yet, it is a prefix of the $y$-root $18 + 15x + 10x^2$). The third solution is a $y$-root of $Q(x, y)$ but not a consistent polynomial (the respective codeword has Hamming distance 15 from $v$). And the fourth solution is a consistent polynomial but does not correspond to the correct codeword.

In this example, the algorithm in Figure 3.4 yields a second polynomial $Q(x, y)$ which is given by

$$Q^{(0)}(x) = 8 + 12x^2 + 9x^3 + 8x^4$$
$$Q^{(1)}(x) = 5 + 14x + 7x^2 + 15x^3 + 4x^4$$
$$Q^{(2)}(x) = 12 + 12x + 15x^2 + 4x^3$$
$$Q^{(3)}(x) = 9 + 10x + 14x^2$$
$$Q^{(4)}(x) = 13 + x$$

and the respective output of $Reconstruct$ is

$$18 + 14x, \ 13 + 9x, \ 10 + x, \ 8 + 8x$$

with only the first and fourth polynomials being $y$-roots of $Q(x, y)$ (as well as being consistent polynomials); the remaining irreducible factor of $Q(x, y)$ is $(13 + x)y^2 + (5 + 18x + 17x^2)y + (18 + 6x + 15x^2)$.

In the example above, the common factors of the two solutions for $Q(x, y)$ correspond to the two (and all) consistent polynomials. However, there are examples where the common $y$-roots in $F_k[x]$ of all the polynomials $Q(x, y)$ generated in Figure 3.2 contain—in addition to the consistent polynomials—also inconsistent ones.

We comment that the connection between the error vector $e$ and the polynomials that appear in the EKE seems to be less obvious than in the classical case; recall that when $\ell = 1$, $e$ can be obtained from $\Lambda(x)$ and $\Omega(x)$ through Chien search [11] and Forney's algorithm [7]. It would be interesting to find such an intimate relationship between $e$ and the polynomials that appear in the EKE also when $\ell > 1$.

### 5.2 Outline for extension

We pointed out that Sudan's algorithm, as well as the decoding algorithm Figure 5.1, can correct more than $(n-k)/2$ errors only when $k \leq (n+1)/3$. Clearly, in many (if not most) practical applications, higher code rates are used. We therefore discuss now the way of extending the procedures in Chapters 3-4 to obtain an efficient implementation of Guruswami-Sudan's algorithm [20], which decodes RS codes beyond the classical decoding radius, for every code rate, as described in Section 2.2.
Then the matrix representation of Guruswami-Sudan’s equations is

\[ \sum_{t=0}^{\ell} \sum_{s=0}^{N_t-1} \binom{t}{s} P_{t-s-s'} y_j^{t-s} = 0, \quad t' = 0, \ldots, r - 1, \quad s' = 0, \ldots, r - 1 - t', \quad j = 1, \ldots, n. \]

Let \( Q \) denote the column vector of unknown coefficients of \( Q(x, y) \), namely

\[ Q = \left( \begin{array}{c} Q^{(0)} \\ Q^{(1)} \\ \vdots \\ Q^{(\ell)} \end{array} \right) \]

where

\[ Q^{(\ell)} = (Q^{(\ell)}_0, Q^{(\ell)}_1, \ldots, Q^{(\ell)}_{N_t-1})^T. \]

Then the matrix representation of Guruswami-Sudan’s equations is

\[ M(r, \ell, n) \cdot Q = 0, \quad (5.1) \]

where \( N \) stands here for the zero column vector of length \( \binom{r+1}{2} n \) and the matrix \( M = M(r, \ell, n) \) is given by

\[ M_{(s', j), (t, s)} = \binom{t}{s} \alpha_j^{t-s-s'} y_j^{t-s} \quad t' = 0, \ldots, r - 1, \quad s' = 0, \ldots, r - 1 - t', \quad j = 1, \ldots, n, \quad t = 0, \ldots, \ell, \quad s = 0, \ldots, N_t - 1. \]

Assume that the column and the row indexes change according to a lexicographic order. Partitioning the rows of \( M \) according to the parameter \( t' \), yields \( r \) submatrices \( M^0, \ldots, M^{r-1} \), where \( M_{(s', j), (t, s)} = M_{(s', j), (t, s)}^t \). The number of rows in \( M^t \) is \( (r - t') n \) and the number of columns is \( \sum_{t=0}^{\ell} N_t \). The conditions on the parameters of the algorithm imply that \( \sum_{t=0}^{r-1} (r - t') n = \binom{r+1}{2} n < \sum_{t=0}^{\ell} N_t \), and thus there exists a nonzero solution \( Q \) to (2.13).
For every $t' = 0, \ldots, r - 1$, we now define a square $(r - t')n \times (r - t')n$ matrix $Y^{t'}$ as follows:

$$Y^{t'}_{(s',j),(s,j)} = (r_{-t'}^i) \beta_j \alpha_j^{i+s'} \quad s' = 0, \ldots, r - 1 - t', \quad j = 1, \ldots, n, \quad i = 0, \ldots, (r - t')n - 1,$$

where $\beta_j \in F \setminus \{0\}$ is any constant that depends on $j$ only. The matrix multiplication $(Y^{t'})^T \cdot M^{t'} \cdot Q$, for $t' = 0, \ldots, r - 1$, results in the following set of $(r_{+1}^n)$ equations:

$$\sum_{t''=0}^{t'} \sum_{s=0}^{N_{t'}^1-1} Q^{(t')_s} \sum_{j=1}^{n} \sum_{s'=0}^{r_{-t'}^i} Y^{t'}_{(s',j),(t',s)} M^{t'}_{(s',j),(t',s)} = 0 \quad i = 0, \ldots, (r - t')n - 1 \quad t'' = 0, \ldots, r - 1,$$

which is equivalent to

$$\sum_{t''=0}^{t'} \sum_{s=0}^{N_{t'}^1-1} Q^{(t')_s} (t')_s^{i+s'} \sum_{j=1}^{n} \beta_j \alpha_j^{i+s'} v_{j}^{t'-t''} = 0 \quad i = 0, \ldots, (r - t')n - 1 \quad t'' = 0, \ldots, r - 1,$$

and to

$$\sum_{t''=0}^{t'} \sum_{s=0}^{N_{t'}^1-1} Q^{(t')_s} S^{t',t''}_{s,i} = 0 \quad i = 0, \ldots, (r - t')n - 1 \quad t'' = 0, \ldots, r - 1,$$

where

$$S^{t',t''}_{s,i} = (t')_s^{i+s'} \sum_{j=1}^{n} \beta_j \alpha_j^{i+s'} v_{j}^{t'-t''}.$$  

(5.2)

For a given $t$ and $t'$, the matrix $S^{t,t'} = [S^{t',t''}_{s,i}]_{i,s}$ is a Hankel matrix, because $S^{t',t''}_{s,i} = S^{t',t''}_{i,s}$. We can therefore denote these ‘syndrome’ elements by $S^{t',t''}_{s,i}$.

It is clear that every bivariate polynomial $Q(x, y)$ which solves (2.13) must solve (5.2) as well. The converse direction is implied by observing that for every $t' \in \{0, 1, \ldots, r - 1\}$, the matrix $Y^{t'}$ is nonsingular. The proof of this statement is omitted.

Similarly to the syndrome computation in Chapter 3, the computation of the syndrome elements can be done in time complexity $O(\ell r^2 n \log^2 n)$. This statement is based on the result in [24] and uses the following two observations:

1. Multiplying the vector $(\beta_j v_{j}^{t'-t''})_{j=1}^{n}$ by the square matrix $[\alpha_j^{i+s}]_{j=1}^{n}$ or by its transpose, where $s$ is any fixed integer, takes $O(n \log^2 n)$ operations.

2. Given $t, t'$, the number of elements $S^{t',t''}_{s,i}$ that have to be computed is less than $2rn$, because $i + s \leq (r - t')n + N_t \leq (r - t')n + r(n - \tau) < 2rn$.

The matrix representation of (5.2) can be considered as consisting of $r$ ‘bands’, where the $t'$th band is divided into $\ell + 1$ Hankel matrices $S^{0,t'}, S^{1,t'}, \ldots, S^{\ell,t'}$ of respective dimensions $(r - t')n \times N_0, (r - t')n \times N_1, \ldots, (r - t')n \times N_\ell$:

$$\begin{pmatrix}
S^{0,0} & S^{1,0} & \ldots & S^{\ell,0} \\
S^{0,1} & S^{1,1} & \ldots & S^{\ell,1} \\
\vdots & \vdots & \ddots & \vdots \\
S^{0,r-1} & S^{1,r-1} & \ldots & S^{\ell,r-1}
\end{pmatrix} \cdot Q = 0.$$  

(5.4)

53
We claim that such an equation set can be solved by quadratic time procedures which are similar to those suggested in Chapter 3. If an FIA-based solution is desired, then it should be applied to the matrix which is obtained from the matrix in (5.4) after some preprocessing: the columns are interleaved in a way similar to that described in Section 3.4, and then the rows are interleaved so that a row taken from band $t'$, where $t' > 0$, always follows a row taken from band $t' - 1$. We mention that interleaving the rows of different Hankel matrices has been suggested in [13] in order to obtain an efficient FIA-based algorithm for decoding cyclic codes. Since the number of unknowns here is $r^2n$ (in comparison with $O(\tau)$ unknowns in the problem we solve in Section 3.4), we get that the time complexity of the FIA-based procedure in our case is $O(\ell^4n^2)$ and its space complexity is $O(r^4n^2)$. The proofs are similar to their counterparts in Section 3.4.

If an extension of the algorithm in Section 3.5 is desired, then the syndrome elements should be considered as located in $r$ different two-dimensional arrays corresponding to the different values taken by $t'$. In each array, there are $\ell + 1$ rows corresponding to the different values taken by $t$. This structure is similar to the one described in Sakata’s recent paper [42]. However, as indicated in Section 3.5, a polynomial solution $T(x, y)$ that is obtained in our method is required to generate the syndrome elements of a single row in each of the $r$ arrays, where a solution polynomial in Sakata’s method generates all the elements in the various $r$ arrays. The time complexity of the solution here is $O(\ell^4n^2)$, which is similar to the result claimed by Sakata in [42]. In our method, $r$ bivariate polynomials are needed for updating the $\ell + 1$ different polynomials computed by the algorithm. The number of coefficients of each of these $\ell + 1 + \tau$ polynomials is $O(r^2n)$, so the space complexity in this case in $O(\ell r^2n)$.

5.3 Preprocessing Sudan’s equations—a second viewpoint

Following the approach in Section 5.2, we derive in this section the homogenous linear equations (3.17) directly from Sudan’s equations (2.6) without going through the EKE. We also explain how the multipliers $\beta_j$ in (5.3) should be chosen so that $n - \tau$ equations, as well as $n - \tau$ unknowns, become redundant.

When $r = 1$, which is the case in Sudan’s algorithm, the row index of the matrix $M = M(1, \ell, n)$ can simply be taken to be $j$ since $s' = \ell' = 0$, and the matrix is given by

$$(M)_{j,(t,s)} = \alpha_j^v_{j_0} \quad j = 1, \ldots, n, \quad t = 0, \ldots, \ell, \quad s = 0, \ldots, N_t - 1.$$  

The equation set (5.1) becomes

$$M(1, \ell, n) \cdot Q = 0, \quad (5.5)$$

where the number of equations is $n$. Denote the number of variables in (5.5) by $n'$. Thus, $n' > n \geq \text{rank}(M)$.

Let $M'$ be the submatrix of $M$ that consists of the leftmost $N_0$ columns (corresponding to $(t, s) = (0, 0), (0, 1), \ldots, (0, N_0 - 1)$), and let $M''$ be the submatrix consisting of the remaining (the rightmost $n - N_0$) columns of $M$. The submatrix $M'$ is independent of the values $v_j$ (received word) and may also form an $n \times N_0$ contiguous submatrix of an $n \times n$ transposed
Vandermonde matrix \( (V^T)_{j,i} = \alpha_{j,i} \). In other words, \( M' \) is a generator matrix of an \([n, N_0]\) RS code with code locators \( \alpha_1, \ldots, \alpha_n \). A respective \((n - N_0) \times n\) parity check matrix \( \tilde{H} \) is given by
\[
(\tilde{H})_{i,j} = \eta_j \alpha_{i,j}, \quad i = 0, \ldots, n - N_0 - 1, \quad j = 1, \ldots, n.
\]
The linearly independent rows of \( \tilde{H} \) are orthogonal to the columns of \( M' \), and \( \text{rank}(M') + \text{rank}(\tilde{H}) = n \).

Let \( \tilde{H} \) multiply each side of (5.5) from the left:
\[
\tilde{H} \cdot M \cdot Q = \tilde{H} \cdot M' \cdot Q^{(0)} + \tilde{H} \cdot M'' \cdot \left( \begin{array}{c} Q^{(1)} \\ \vdots \\ Q^{(t)} \end{array} \right) = \tilde{H} \cdot M'' \cdot \left( \begin{array}{c} Q^{(1)} \\ \vdots \\ Q^{(t)} \end{array} \right) = 0. \tag{5.6}
\]
We thus get the following set of \( n - N_0 \) homogeneous linear equations in the unknown coefficients of \( Q^{(1)}(x), \ldots, Q^{(t)}(x) \) only
\[
\sum_{i=1}^{n} \sum_{s=0}^{N_t-1} Q^{(s)}(t) \eta_j \alpha_{i+s,j} x^t = 0, \quad i = 0, 1, \ldots n - N_0 - 1. \tag{5.7}
\]
The rank of the system (5.7) is \( \min\{\text{rank}(M), \text{rank}(\tilde{H}) = n - N_0\} \). The number of unknowns in (5.7) is \( n' - N_0 \).
We examine two cases:

1. If \( \text{rank}(M) \geq \text{rank}(\tilde{H}) \), then the number of free variables in (5.7) is \((n' - N_0) - (n - N_0) = n' - n \). On the one hand, \( n' - n \geq 1 \). On the other hand, \( n' - n \leq n' - \text{rank}(M) \).

2. If \( \text{rank}(M) < \text{rank}(\tilde{H}) \), then the number of free variables in (5.7) is \((n' - N_0) - \text{rank}(M) \).
   On the one hand, \( n' - N_0 - \text{rank}(M) > n' - N_0 - (n - N_0) = n' - n \geq 1 \). On the other hand, \( n' - N_0 - \text{rank}(M) < n' - \text{rank}(M) \).

We conclude that the number of free variables in (5.7) is always positive and not greater than the number of free variables in (5.5). Hence, there exists a nontrivial solution for (5.7), while no information is lost by the transition from (5.5) to (5.7).

The system (5.7) can be rewritten as
\[
\sum_{i=1}^{n} \sum_{s=0}^{N_t-1} Q^{(s)}(t) \cdot \left( \sum_{j=1}^{n} \eta_j \alpha_{i+s,j} x^t \right) = 0, \quad i = 0, 1, \ldots n - N_0 - 1.
\]
Using the definition (3.13), namely
\[
S^{(t)}_{i+s} = \sum_{j=1}^{n} \eta_j \alpha_{i+s,j} x^t,
\]
we obtain the set of homogeneous linear equations (3.17):
\[
\sum_{i=1}^{n} \sum_{s=0}^{N_t-1} Q^{(s)}(t) S^{(t)}_{i+s} = 0, \quad i = 0, 1, \ldots n - N_0 - 1,
\]
which was derived from the EKE in Section 3.3. Note that the number of equations here is \( n - N_0 \), which, by (2.1) and (2.3), equals the number \( \tau \) of correctable errors in Sudan's
algorithm. The syndrome formula here is a special case of (5.3), where \( r = 1 \), \( t' = 0 \), and \( \beta_j = \eta_j = 1 / (\prod_{i \neq j} (\alpha_i - \alpha_j)) \).

The syndrome vector \((S^{(t)})_{s=0}^{n-2-t(k-1)}\) can be though of as obtained by the matrix-vector multiplication

\[
H^{(t)} \cdot \begin{pmatrix}
v_1^t \\
v_2^t \\
\vdots \\
v_t^t
\end{pmatrix},
\]

where the matrix \(H^{(t)}\) is defined by

\[
H^{(t)} = \left[ \eta_j \alpha_j^s \right]_{s,j} \quad s = 0, 1, \ldots, n-2-t(k-1), \quad j = 1, 2, \ldots, n.
\]

We point out that \(H = \tilde{H}^{(1)}\) is an \((n - k) \times n\) parity check matrix of the original code \(C\), where \(\tilde{H}^{(2)}, \ldots, \tilde{H}^{(t)}\), as well as \(\tilde{H}\), may be considered as consisting of contiguous rows of \(H\).

Once we have a solution for \(Q^{(1)}, \ldots, Q^{(t)}\), we substitute it in (5.5) and get the following system of \(n\) non-homogeneous linear equations in the \(N_0\) unknown components of \(Q^{(0)}\).

\[
M' \cdot Q^{(0)} = -M'' \cdot \begin{pmatrix}
Q^{(1)} \\
\vdots \\
Q^{(t)}
\end{pmatrix}.
\]

By (5.6), the right-hand side of (5.8) is orthogonal to the linear subspace spanned by the rows of \(\tilde{H}\). Since the columns of \(M'\) form a basis of the dual subspace, the right-hand side of (5.8) can be written as a unique linear combination of the columns in \(M'\). Consequently, there is a unique solution for \(Q^{(0)}\) in (5.8). Moreover, the vector \(Q^{(0)}\) can be found efficiently through interpolation, as explained in Section 3.6.
Chapter 6

Lower bounds on the list decoding radius of general block codes

In this chapter, lower bounds are derived on the list decoding radius $\Delta_{\ell}(n, d)$ of general $(n, M, d)$ codes. The terminology and definitions from Section 1.5 are used throughout this chapter. In Section 6.1, the GS bound is shown to hold for every block code, thus generalizing Guruswami-Sudan’s result (summarized in Theorem 1.1). In Section 6.2, sufficient conditions for the existence of failing lists (see the definition in Section 1.5.1) are given by means of a combinatorial structure referred to as $(\ell, r)$-configuration. These sufficient conditions are later used in Chapter 7 in order to obtain upper bounds on the decoding radius $\Delta_{\ell}^{RS}(n, d)$ of $[n, k, d]$ RS codes.

However, in Section 6.3, the $(\ell, r)$-configuration is shown to be the only combinatorial structure which is possible for a failing list whenever the parameters $\ell, n, d$ satisfy certain divisibility constraints. Such a characterization of the failing list structure is used in Sections 6.3–6.4 to obtain lower bounds on $\Delta_{\ell}(n, d)$ which improve on the GS bound.

Throughout this chapter, we fix the alphabet $F$ of size $q$, the length $n$ and minimum Hamming distance $d < n$ of an $(n, M, d)$ code over $F$, and a list size $\ell$. We let $r$ be the unique integer such that $1 - d/n \in [\rho_r, \rho_{r+1})$, and we use the notation $[n]$ for the set $\{1, 2, \ldots, n\}$.

6.1 Generalization of the GS bound

For the sake of completeness, we repeat here Theorem 1.2.

Theorem 1.2 Let $(\ell, n, d, q)$ be an admissible quadruple. Then

$$\Delta_{\ell}(n, d; q) \geq [\tau_{\ell}(n, d)] - 1,$$

where $\tau_{\ell}(n, d)$ is defined by (1.13).

Proof: Assume to the contrary that there is an $(n, M, d)$ code $C$ for which $\Delta_{\ell}(C) < [\tau_{\ell}(n, d)] - 1$. It follows that there is a set of $\ell+1$ codewords, $\mathcal{L} = \{c_0, c_1, \ldots, c_{\ell}\} \subseteq C$ and a word $v \in F^n$ such that $d_H(c_s, v) < [\tau_{\ell}(n, d)]$ for every $0 \leq s \leq \ell$. 

57
For every $\mu \in [n]$, denote by $x_\mu$ the number of words in $L$ that agree with $v$ on the $\mu$th position. On the one hand, it is clear that

$$\sum_{\mu=1}^{n} x_\mu > (\ell+1)(n - \tau_\ell(n, d)) \quad (6.1)$$

On the other hand, the number of different (unordered) pairs $\{c_s, c_t\} \subseteq L$ that agree on their $\mu$th coordinate is at least $\binom{x_\mu}{2}$. Since $d_H(c_s, c_t) \geq d$ for every $0 \leq s < t \leq \ell$, it follows that the total number of agreement coordinates, when ranging over all pairs $\{c_s, c_t\} \subseteq L$, cannot exceed $\binom{(\ell+1)(n-d)}{2}$; therefore,

$$\sum_{\mu=1}^{n} \binom{x_\mu}{2} \leq \binom{(\ell+1)(n-d)}{2} \quad (6.2)$$

Define

$$y = \frac{1}{r} \left( \binom{(\ell+1)(n-d)}{2} - \binom{t}{2}n \right) \quad (6.3)$$

By the definition of the parameter $r$, we get that $0 \leq y < n$. It can be easily verified that the right-hand side of (6.1) equals $rn+y$. Denote $t = \lfloor y \rfloor + 1$. It follows from (6.1) that

$$\sum_{\mu=1}^{n} x_\mu \geq \frac{1}{r} \left( n + rt \right)$$

$$(6.4)$$

Regard $x_1, x_2, \ldots, x_n$ as integer variables that are constrained to satisfy (6.4). By [27, p. 526], the minimum of the sum $\sum_{\mu=1}^{n} \binom{x_\mu}{2}$ is

$$\frac{1}{2} (t(r+1)^2 + (n-t)r^2 - (rn+t)) = \binom{t}{2}n + rt > \binom{t}{2}n + ry = \binom{(\ell+1)}{2}(n-d)$$

contradicting (6.2).

6.2 $(\ell, r)$-configurations

We define an $(\ell, r)$-configuration as a set $L$ of $\ell+1$ words in $F^n$ such that for every position $\mu \in [n]$ the following holds: there are exactly $r$ or $r+1$ words in $L$ that agree on that position by taking the same value, $c_\mu$, and the remaining $\ell+1-r$ (respectively, $\ell-r$) words are all distinct on that position, neither does any of them take there the value $c_\mu$.

We now repeat the definition with slight more detail. Let $S_1, S_2, \ldots, S_{\binom{(\ell+1)}{r}}$ be all the distinct subsets of $\{0, 1, \ldots, \ell\}$ of size $r$, and let $S'_1, S'_2, \ldots, S'_{\binom{(\ell+1)}{r+1}}$ be all the distinct subsets of $\{0, 1, \ldots, \ell\}$ of size $r+1$. A partition vector of $[n]$ is an (ordered) list of $\binom{(\ell+2)}{r+1} = \binom{(\ell+1)}{r} + \binom{(\ell+1)}{r+1}$ disjoint subsets,

$$\left( I_1, I_2, \ldots, I_{\binom{(\ell+1)}{r}}, I'_1, I'_2, \ldots, I'_{\binom{(\ell+1)}{r+1}} \right)$$

whose union is $[n]$. A partition vector $P$ is said to be proper if $I'_j = \emptyset$ for all $j$. The existence of an $(\ell, r)$-configuration over $F$ implies $q \geq \ell+2-r$, whenever there is a nonempty partition element $I_i$. 58
We will hereafter abbreviate notations and write \((I_i)_i\| (I'_j)_j\) for a partition vector; a proper partition vector will also be written as \((I_i)_i\). Given a partition vector \(\mathcal{P} = (I_i)_i\| (I'_j)_j\) an \((\ell, r)\)-configuration with respect to \(\mathcal{P}\) is a set of \(\ell+1\) words \(\mathcal{L} = \{c_0, c_1, \ldots, c_{\ell}\} \subseteq F^n\) that satisfies the following two conditions:

- For every \(i = 1, 2, \ldots, (r\ell^{-1})\), the words in \(\mathcal{L}_i = \{c_s\}_{s \in S_i}\) are identical on the positions indexed by \(I_i\), while none of the words in \(\mathcal{L} \setminus \mathcal{L}_i\) agrees on any of those positions with any other word in \(\mathcal{L}\).
- The same as the previous condition, with \(I'_j\) replacing \(I_i\) and \(\mathcal{L}'_j = \{c_s\}_{s \in S'_j}\) replacing \(\mathcal{L}_i\) for \(j = 1, 2, \ldots, (r\ell^{-1})\).

The existence of an \((\ell, r)\)-configuration \(\mathcal{L}\) with respect to a partition vector \(\mathcal{P} = (I_i)_i\| (I'_j)_j\) implies the existence of an incidence structure \(D(\mathcal{L}) = (\mathcal{L}, \mathcal{B}, \mathcal{M})\) (see [6, Ch. 1]) with \(\ell+1\) ‘points,’ corresponding to the codewords in \(\mathcal{L}\), and a multi-set \(\mathcal{B}\) of \(n\) (not necessarily distinct) ‘blocks.’ The \((\ell+1) \times n\) incidence matrix \(\mathcal{M}\), which represents the incidence relation, is defined as follows:

\[
\mathcal{M}_{s, \mu} = \begin{cases} 
1 & \text{if } \mu \in I_s \text{ and } s \in S_i, \text{ for some } i \\
1 & \text{if } \mu \in I'_s \text{ and } s \in S'_j, \text{ for some } j \\
0 & \text{otherwise}
\end{cases} \quad s \in \{0, 1, \ldots, \ell\}, \quad \mu \in [n].
\]

Using the terminology in [6], \(D(\mathcal{L})\) is an incidence structure with possibly repeated blocks and up to two block sizes, \(r\) and \(r+1\); when the partition elements \(I_i\) and \(I'_j\) are all of size \(\leq 1\), no repeated blocks appear, and when \(\mathcal{P}\) is a proper partition vector, only the block size \(r\) is allowed.

Lemma 6.1 below provides sufficient conditions for an \((\ell, r)\)-configuration \(\mathcal{L}\) to form a failing list.

**Lemma 6.1** Let \(\mathcal{L} = \{c_0, c_1, \ldots, c_{\ell}\}\) be an \((\ell, r)\)-configuration with respect to a partition vector \((I_i)_i\| (I'_j)_j\) of \([n]\) that satisfies both

\[
\sum_{i : \{s, t\} \subseteq S_i} |I_i| + \sum_{j : \{s, t\} \subseteq S'_j} |I'_j| \leq n - d, \quad 0 \leq s < t \leq \ell, \quad (6.5)
\]

and

\[
\sum_{i : s \in S_i} |I_i| + \sum_{j : s \in S'_j} |I'_j| \geq n - \lceil \tau_{\ell}(n, d) \rceil, \quad 0 \leq s \leq \ell. \quad (6.6)
\]

Then \(\mathcal{L}\) is a failing list: each word in \(\mathcal{L}\) is at Hamming distance at most \(\lceil \tau_{\ell}(n, d) \rceil\) from the majority-vote word \(v \in F^n\) that agrees on any position in \(I_i\) (respectively, \(I'_j\)) with the words in \(\mathcal{L}_i\) (respectively, \(\mathcal{L}'_j\)).

**Proof:** By (6.5), every two words in \(\mathcal{L}\) agree on at most \(n-d\) positions and, thus, \(d_{H}(c_s, c_t) \geq d\) for every \(0 \leq s < t \leq \ell\). In addition, by (6.6), the Hamming distance of each word in \(\mathcal{L}\) from the word \(v\) is not greater than \(\lceil \tau_{\ell}(n, d) \rceil\). It follows that \(\mathcal{L}\) is a failing list.
The following corollary describes a case with certain symmetry where Lemma 6.1 can be applied. This special case is later used to indicate RS codes that contain failing lists.

**Corollary 6.2** Suppose there are integers $\gamma > 0$ and $\gamma' \geq 0$ that satisfy
\[
\binom{\ell+1}{r} \gamma + \binom{\ell+1}{r+1} \gamma' = n \quad \text{and} \quad \binom{\ell-1}{r-2} \gamma + \binom{\ell-1}{r-1} \gamma' = n-d
\]
(here $\tau(n,d)$ is an integer and its value is given by $\binom{\ell}{r} \gamma + \binom{\ell}{r+1} \gamma'$). Let $\mathcal{L} = \{c_0, c_1, \ldots, c_\ell\}$ be an $(\ell, r)$-configuration with respect to a partition vector $(I_i)_{i\in\mathbb{Z}}$ of $V$ where $|I_i| = \gamma$ and $|I'_i| = \gamma'$. Let $\nu \in F^n$ be the majority-vote word. Then $\mathcal{L}$ is a failing list in which $d_H(c_s, c_t) = d$ for every $0 \leq s < t \leq \ell$ and $d_H(c_s, \nu) = \tau(n, d)$ for every $0 \leq s \leq \ell$.

We point out that the failing lists described in [23], corresponding to cases where the relative minimum distance is $1 - \rho_\gamma$, have a combinatorial structure which is a special case of the $(\ell, r)$-configuration in Corollary 6.2, obtained when $\gamma' = 0$. As indicated in [23], the incidence structure $D(\mathcal{L})$ in this case is a replication of the trivial (complete) balanced incomplete block design (in short, BIBD) with parameters $(\ell+1, r, (n-d)/\gamma)$. In such a BIBD, the $n=\binom{\ell+1}{r}$ blocks correspond to all the distinct $r$-subsets of the point set $\mathcal{L}$, each pair of points appears in exactly $(n-d)/\gamma = \binom{\ell}{r}$ blocks, and each single point appears in exactly $(n-\tau(n,d))/\gamma = \binom{\ell}{r-1}$ blocks.

**Example 6.1** Figure 6.1 presents a $(3,2)$-configuration of four words of length 10 over $GF(11)$ and the respective majority-vote word $\nu$. The words $c_0, c_1, c_2, c_3$ are codewords of a $[10,4,7]$ RS code whose code locators are 0,5,6,4,2,1,7,3,9,8. The configuration forms a failing list since every two codewords agree on $n-d = 3$ positions and $\nu$ agrees with every codeword on $\tau_3(10,7) = 4$ positions. Note that for every two distinct $s, t \in \{0,1,2,3\}$ there is a unique position on which only $c_s$ and $c_t$ agree, and for every three distinct $s, t, u \in \{0,1,2,3\}$ there is a unique position on which only $c_s$, $c_t$, and $c_u$ agree. This list thus corresponds to the structure described in Corollary 6.2, where $\gamma = \gamma' = 1$.

\[
c_0 = 0 0 0 0 0 0 0 0 0 0 \\
c_1 = 0 0 2 3 0 4 4 5 10 1 \\
c_2 = 0 8 0 3 1 0 3 5 6 8 \\
c_3 = 6 0 0 3 8 5 0 1 10 8 \\
\nu = 0 0 0 3 0 0 0 5 10 8
\]

Figure 6.1: $(3,2)$-configuration over $GF(11)$.

**Example 6.2** Figure 6.2 presents a $(4,3)$-configuration of five codewords of a $[10,4,7]$ RS code over $GF(16)$ (the field elements are represented as polynomials over $GF(2)$ modulo $x^4 + x + 1$, and the four polynomial coefficients of each element are written in hexadecimal notation). The rate, $1 - d/n = 3/10$, equals the boundary rate $\rho_5(4)$. This configuration
follows the structure described in Corollary 6.2, where $\gamma = 1$ and $\gamma' = 0$, and it therefore forms a failing list. It can be easily verified that indeed every two codewords agree on $n-d = 3$ positions and $\psi$ agrees with every codeword on $\tau_4(10, 7) = 4$ positions.

The list structure here corresponds to the (complete) BIBD$(5, 3, 3)$ (which has 10 blocks), and we show in the sequel (Lemma 7.14), that, in fact, every failing list of five codewords in a $(10, M, 7)$ code over any alphabet $F$ must have the form of a BIBD$(5, 3, 3)$. It follows that such a failing list cannot be realized over the binary alphabet. In Chapter 7, it is proved that this failing list structure can be realized in RS codes over $GF(q)$ if and only if $q$ is a power of 2 not smaller than 16 (see Propositions 7.4 and 7.5).

$\begin{align*}
c_0 &= 0 0 0 0 0 0 0 0 0 0 \\
c_1 &= 0 0 0 9 f 2 a 4 f b \\
c_2 &= 0 9 1 0 0 b a 4 7 c \\
c_3 &= 0 0 0 5 a 0 a f c \\
c_4 &= 0 0 0 4 f 0 0 1 4 f c \\
\end{align*}$

Figure 6.2: $(4, 3)$-configuration over $GF(16)$.

Fix a list size $\ell$ and a rational number $\delta \in (0, 1]$. We claim that one can always extend $\ell$ to some admissible quadruple $(\ell, n, d, q)$ with $d/n = \delta$, such that $\ell+1$ words that form a failing list are contained in $F^n$ (where $F$ is an alphabet of size $q$). Indeed, replacing $d$ by $n\delta$ in (6.7) (where $r$ is uniquely determined by $\delta$ and $\ell$) transforms (6.7) into a set of two homogeneous equations in the three unknowns $\gamma$, $\gamma'$, and $n$. A nontrivial integer solution must then exist. For any value of $q$ greater than $\ell+2-r$, we can find $\ell+1$ words in $F^n$ that form an $(\ell, r)$-configuration with respect to some partition vector $P = (I_i)_i || (I'_j)_j$ of $[n]$, where $|I_i| = \gamma$ for $1 \leq i \leq \binom{\ell+1}{r}$, and $|I'_j| = \gamma'$ for $1 \leq j \leq \binom{\ell+1}{r+1}$. By Corollary 6.2, this is a failing list.

## 6.3 Necessary conditions on failing lists

Proposition 6.3 below motivates our interest in failing lists that form $(\ell, r)$-configurations. It states that when $\tau_i(n, d)$ is an integer, namely when (1.14) holds, every failing list of size $\ell+1$ is necessarily an $(\ell, r)$-configuration. The sufficient condition for the existence of a failing list, as stated in Lemma 6.1, is thus necessary in cases where (1.14) holds.

It turns out that our necessary conditions imply that there is a range of parameters where the GS bound is not tight for any code. For example, property N3 in Proposition 6.3 below indicates the non-existence of failing lists in cases where the alphabet size is small.

**Proposition 6.3** Let $\ell$, $r$, $n$, and $d$ be integers for which (1.14) holds, and let $L$ be a failing list of size $\ell+1$ that is contained in an $(n, M, d)$ code over $F$.

$N1$ The list $L$ is an $(\ell, r)$-configuration with respect to some partition vector $P = (I_i)_i || (I'_j)_j$ of $[n]$ that satisfies conditions (6.5)–(6.6) with equality.
**N2** \( \mathcal{P} \) is proper (i.e., exactly \( r \) out of the \( \ell+1 \) words in \( \mathcal{L} \) agree on every position) if and only if \( 1 - d/n = r/(r-1)/((\ell+1)) = \rho_r \).

**N3** The alphabet size \( q \) is at least \( \ell+2-r \).

**Proof:** Let \( \mathcal{L} = \{ c_0, c_1, \ldots, c_r \} \) be a failing list with the given parameters, and let \( v \) be the word in \( F^n \) for which \( d_H(c_i, v) \leq \tau = \tau_r(n, d) \) for every \( s \in \{ 0, 1, \ldots, \ell \} \). As in the proof of Theorem 1.2, we denote by \( x_{\mu} \), \( \mu \in [n] \), the number of words in \( \mathcal{L} \) that agree with \( v \) on the \( \mu \)th position. By arguments similar to those in the proof of Theorem 1.2, we get that

\[
\sum_{\mu=1}^{n} x_{\mu} \geq (\ell+1)(n-\tau) \quad (6.8)
\]

and

\[
\sum_{\mu=1}^{n} \left( \frac{x_{\mu}}{2} \right) \leq \left( \frac{\ell+1}{2} \right)(n-d) \quad (6.9)
\]

Let \( y \) be as in (6.3). Under the assumption that (1.14) holds, \( y \) must be an integer. When \( 1 - d/n = \rho_r \), we get \( y = 0 \); otherwise, \( 0 < y < n \). Regard \( x_1, x_2, \ldots, x_n \) as integer variables that are constrained to satisfy (6.8) with equality. By [27, p. 526], the minimum of the sum \( \sum_{\mu=1}^{n} \left( \frac{x_{\mu}}{2} \right) \) is attained when (and only when) \( y \) of the variables take the value \( r+1 \) while the rest take the value \( r \); such an assignment satisfies (6.9) with equality. Since the minimum could only increase if we constrained the sum \( \sum_{\mu=1}^{n} x_{\mu} \) to be larger, we have thus characterized the only feasible solutions to (6.8)–(6.9).

We now define the partition vector \( \mathcal{P} \) that is stated in the lemma. For every subset \( S_i \) (respectively, \( S_j^r \)) of \( \{ 0, 1, \ldots, \ell \} \) of size \( r \) (respectively, \( r+1 \)), let \( I_i \) (respectively, \( I_j^r \)) be the set of positions on which the words in \( \mathcal{L}_i = \{ c_s : s \in S_i \} \) (respectively, \( \mathcal{L}_j = \{ c_s : s \in S_j^r \} \))—and only these words—agree with \( v \).

Since the union of \( \bigcup_i I_i \) and \( \bigcup_j I_j^r \) is necessarily \([n]\), it follows that \( \mathcal{P} = (I_i)_i \| (I_j^r)_j \) is a partition vector. We have,

\[
\sum_{i: \{s,t\} \subseteq S_i} |I_i| + \sum_{j: \{s,t\} \subseteq S_j^r} |I_j^r| \leq n - d_H(c_s, c_t), \quad 0 \leq s < t \leq \ell, \quad (6.10)
\]

and

\[
\sum_{i: s \in S_i} |I_i| + \sum_{j: s \in S_j^r} |I_j^r| = n - d_H(c_s, v), \quad 0 \leq s \leq \ell. \quad (6.11)
\]

Since \( \mathcal{L} \) is a failing list, we can bound the right-hand side of (6.10) from above by \( n-d \) and the right-hand size of (6.11) from below by \( n-\tau \). This, in turn, implies that conditions (6.5)–(6.6) hold. Furthermore, since (6.8)–(6.9) hold with equality, we obtain,

\[
\frac{1}{2} \sum_{0 \leq s < t \leq \ell} \left( \sum_{i: \{s,t\} \subseteq S_i} |I_i| + \sum_{j: \{s,t\} \subseteq S_j^r} |I_j^r| \right) = \sum_{\mu=1}^{n} \left( \frac{x_{\mu}}{2} \right) = \left( \frac{\ell+1}{2} \right)(n-d)
\]

and

\[
\sum_{s=0}^{\ell} \left( \sum_{i: s \in S_i} |I_i| + \sum_{j: s \in S_j^r} |I_j^r| \right) = \sum_{\mu=1}^{n} x_{\mu} = (\ell+1)(n-\tau).
\]
It follows that conditions (6.5)–(6.6) hold with equality, and so does (6.10). The equality in (6.10) implies that when \( x_\mu = r \) (respectively, \( x_\mu = r+1 \)), there are exactly \( \binom{r}{2} \) (respectively, \( \binom{r+1}{2} \)) different pairs of words in \( \mathcal{L} \) that agree on their \( \mu \)th coordinate. In particular, a word in \( \mathcal{L} \setminus \mathcal{L}_i \) (respectively, \( \mathcal{L} \setminus \mathcal{L}'_j \)) does not agree on any position in \( I_i \) (respectively, \( I'_j \)) with any other word in \( \mathcal{L} \).

We conclude that \( \mathcal{L} \) is an \((\ell, r)\)-configuration with respect to the partition vector \( \mathcal{P} \), and property N1 is thus proved. Recalling that \( y=0 \) when \( 1-d/n = \rho_r \), property N2 is proved as well. Since \( n-y > 0 \), at least one partition element \( I_i \) is nonempty. If \( \mu \in S_1 \), then the number of different alphabet symbols represented in the \( \mu \)th components of \( c_0, \ldots, c_\ell \) is at least \( \ell+2-r \). Property N3 is thus implied.

Based on Proposition 6.3, Proposition 6.4 below provides necessary conditions on the existence of failing lists by means of constant-weight codes. If \( F \) is an additive group, then an \((n, d, w)\) constant-weight code over \( F \) is a subset of \( F^n \) such that the Hamming weight (i.e., the number of nonzero components) of every codeword is \( w \) and the minimum Hamming distance between different codewords is \( d \) (see also [27, page 524]).

**Proposition 6.4** Let \( \ell, r, n, \) and \( d \) be as in Proposition 6.3. Suppose that a failing list is contained in some \((n, M, d)\) code \( \mathcal{C} \) over an additive group \( F \) of size \( q \). Then there is some failing list in \( F^n \) that forms an \((n, d, \tau_\ell(n, d))\) constant-weight code \( \tilde{\mathcal{C}} \) over \( F \), consisting of \( \ell+1 \) codewords. The Hamming distance between different codewords in \( \tilde{\mathcal{C}} \) is exactly \( d \).

**Proof:** Let \( \mathcal{L} = \{c_i\}_{i=0}^{\ell} \) be the failing list in \( \mathcal{C} \), and let \( v \) be as in the proof of Proposition 6.3. By property N1 of that proposition, the set \( \{c_0-v, c_1-v, \ldots, c_\ell-v\} \) forms the required constant-weight code \( \tilde{\mathcal{C}} \) over \( F \).

### 6.4 Block designs and failing lists

Proposition 6.5 below deals with list sizes \( \ell \geq n-1 \). In particular, it states that when \( \ell = n-1 \), the GS bound can be attained only when there is a symmetric BIBD with parameters \((n, r, n-d)\). Such a design consists of the same number \( n \) of ‘points’ and ‘blocks’. The block size is \( r \) and each pair of distinct points appears in exactly \( n-d \) blocks (see [6], [21, Ch. 10], and [27, Section 2.5]).

**Proposition 6.5** Let \( \ell, r, n, d, q \) be as in Proposition 6.3.

**B1** If \( n = \ell+1 \) then \( \Delta_\ell(n, d; q) = \tau_\ell(n, d) - 1 \) only when there is a BIBD \((n, r, n-d)\) with \( r(r-1) = (n-1)(n-d) \).

**B2** If \( n < \ell+1 \) then \( \Delta_\ell(n, d; q) \geq \tau_\ell(n, d) \).

(In contrast, Justesen and Høholdt identify triples \((\ell, n, d)\) for which the existence of the corresponding BIBD implies the tightness of the GS bound for MDS codes over sufficiently large fields; see (the proof of) Theorem 4 in [23].)

The proof of Proposition 6.5 is given towards the end of this section and uses the analysis below. Let \( \mathcal{L} \) be a failing list as in Proposition 6.3. We consider the incidence structure
\( D(\mathcal{L}) = (\mathcal{L}, \mathcal{B}, \mathcal{M}) \) as a generalization of a BIBD\((\ell+1, r, n-d)\), referred to as a quasi-BIBD and denoted QBIBD\((\ell+1, r, n-d; n)\). Similarly to a BIBD, every pair of points appears in exactly \( n-d \) blocks (the incidence structure is pairwise balanced), and each single point appears in exactly \( n-\tau_\ell(n, d) \) blocks. However, in a QBIBD, \( y \) blocks are of size \( r+1 \), where \( y \) is defined by (6.3), and the remaining \( n-y \) blocks are of size \( r \). In addition, repeated blocks are allowed in a QBIBD.

Note that the number of blocks \( n \) appears as a parameter in the definition of a QBIBD since it is not uniquely determined by the other three parameters. However, the following connection between the parameters must hold:

\[
\binom{r}{2} n \leq \binom{\ell+1}{2} (n-d) < \binom{\ell+1}{2} n. \tag{6.12}
\]

When the left inequality in (6.12) holds with equality (i.e., the code relative minimum distance is \( 1 - \rho_r \)), the \( n \) blocks are all of size \( r \).

Some useful properties of a BIBD, such as Fisher’s inequality (see, for example, [6, p. 81]), hold also for a QBIBD, as stated in the following lemma. The proof is essentially the same as in the case of a BIBD, and it is included for the sake of completeness.

**Lemma 6.6** In a QBIBD\((\ell+1, r, n-d; n)\), there are at least \( \ell+1 \) distinct blocks. In particular, \( \ell+1 \leq n \).

**Proof:** Let \( D(\mathcal{L}) = (\mathcal{L}, \mathcal{B}, \mathcal{M}) \) be an incidence structure of a QBIBD\((\ell+1, r, n-d; n)\). The entries of the \((\ell+1) \times (\ell+1)\) matrix \( \mathcal{M}\mathcal{M}^T \) are given by

\[
(M\mathcal{M}^T)_{s,t} = \begin{cases} n-\tau_\ell(n, d) & \text{if } s = t \\ n-d & \text{if } s \neq t \end{cases},
\]

and so,

\[
\det \mathcal{M}\mathcal{M}^T = (d-\tau_\ell(n, d))^\ell \cdot (\ell(n-d)+n-\tau_\ell(n, d)) \neq 0.
\]

(To verify the inequality above, recall that \( \tau_\ell(n, d) < d \leq n \).) It follows that \( \text{rank} \mathcal{M} \geq \text{rank} \mathcal{M}\mathcal{M}^T = \ell+1 \), so \( \mathcal{M} \) contains at least \( \ell+1 \) linearly independent—and hence distinct—columns.

The following corollary is implied by Proposition 6.3 and Lemma 6.6.

**Corollary 6.7** Let \( \ell, r, n, \) and \( d \) be integers for which (1.14) holds. Then a failing list of size \( \ell+1 \) is contained in an \((n, M, d)\) code over some alphabet \( F \) only if there exists a QBIBD\((\ell+1, r, n-d; n)\). In particular, \( \ell+1 \leq n \) whenever a failing list \( \mathcal{L} \) exists.

The next lemma deals with the special case where \( n = \ell+1 \) and is used in the proof of Proposition 6.5.

**Lemma 6.8** A QBIBD\((n, r, n-d; n)\) is a (symmetric) BIBD\((n, r, n-d)\).

64
**Proof:** By Lemma 6.6, the \( n \) blocks of a QBIBD\((n, r, n-d; n)\) are all distinct. Now, each point appears in \( n-\tau_t(n,d) \) blocks and, so, \( \tau_t(n,d) \) is an integer. The divisibility condition (1.14), which necessarily holds here, becomes

\[
2r \quad \text{divides} \quad r(r+1) + (n-1)(n-d).
\]

By (6.12), we also require

\[
r(r-1) \leq (n-1)(n-d) < r(r+1).
\]

The above two constraints are satisfied only if \( (n-1)(n-d) = r(r-1) \), implying that the \( n \) distinct blocks are all of the same size \( r \). The QBIBD is thus a BIBD. \( \blacksquare \)

**Proof of Proposition 6.5:** Combine Corollary 6.7 and Lemma 6.8. Necessary conditions on the parameters of a symmetric BIBD were given by Bruck, Chowla, and Ryser (see [6, page 100] or [21, page 133]). It follows from Proposition 6.5 that whenever these conditions are not satisfied by \( (n, r, n-d) \), no \( (n, M, d) \) code attains the GS bound with equality. For example, since there is no BIBD\((22, 7, 2)\), we obtain for every alphabet size \( q \),

\[
\Delta_{21}(22, 20; q) \geq 15 = \tau_{21}(22, 20).
\]

Similarly,

\[
\Delta_{42}(43, 42; q) \geq 36 = \tau_{42}(43, 42).
\]
Chapter 7

Bounds on the list decoding radius of Reed-Solomon codes

In this chapter, upper and lower bounds are derived on the list decoding radius $\Delta^\text{RS}_\ell(n, d)$ of $[n, k, d]$ RS codes. The results, without their proofs, are presented in Section 7.1. The GS lower bound of Theorem 1.1 is proved as tight (namely, as an upper bound on $\Delta^\text{RS}_\ell(n, d)$) in certain cases described in Propositions 7.1-7.4. Lower bounds which improve on the GS bound are given in Propositions 7.5 and 7.6.

Tools for analyzing the list decoding radius of RS codes are developed in Section 7.2, based on the combinatorial characterization of failing lists made in Chapter 6. These tools are then used in Section 7.3 to prove Propositions 7.1-7.6.

Throughout this chapter we use the term rate of an $[n, k, d]$ RS code to denote the ratio $k/n = \frac{k}{n}$, which equals $1 - d/n$. The intervals $[\rho_r, \rho_{r+1})$ defined in Section 1.5.1 are therefore referred to as rate intervals.

7.1 Main Results

For triples $(\ell, n, k)$ that correspond to the first and last sub-intervals in (1.11) (specifically, $(k-1)/n \leq 2/(\ell(\ell+1))$ or $(k-1)/n \geq 1 - (2/(\ell+1)))$, we find a variety of finite fields $\text{GF}(q)$ over which there are $[n, k, d]$ RS codes that attain the GS bound. These results are summarized in Propositions 7.1 and 7.2 below.

Proposition 7.1 covers the high-rate range (i.e., small values of $d/n$) and identifies quadruples $(\ell, n, d, q)$ for which a list-$\ell$ decoder for the worst $[n, k, d]$ RS code, and hence for the worst $(n, M, d)$ code, does no better than a list-1 (‘classical’) decoder.

**Proposition 7.1** Let the RS-admissible quadruple $(\ell, n, d, q)$, other than $(3, 2, 1, 2)$, satisfy

$$d/n \leq \frac{2}{\ell+1} \left( 1 - \rho_1 \right).$$

Assume in addition that when $d > 1$, the integer $\lceil (d-1)/2 \rceil$ divides either $q-1$ or $q$. Then,

$$\Delta^\text{RS}_\ell(n, d; q) = \lceil \tau_\ell(n, d) \rceil - 1 = \lfloor (d-1)/2 \rfloor = \Delta^\text{RS}_1(n, d; q).$$
In particular, for $2 \leq d \leq 5$ and every field size $q$,
\[
\Delta^\text{RS}_\ell(n, d; q) = \left\lfloor \frac{(d-1)/2}{1} \right\rfloor, \quad 1 \leq \ell \leq \frac{2n}{d} - 1.
\]

Proposition 7.2 covers the low-rate range (the leftmost sub-interval in (1.11), namely high values of $d/n$) and makes use of the following definition. A subset $X$ of an Abelian group is called a weak Sidon set if every four distinct elements $\theta_1, \theta_2, \theta_3, \theta_4 \in X$ satisfy $\theta_1 + \theta_2 \neq \theta_3 + \theta_4$ (see [3], [9], [19], [39]). The notation $\mathbb{Z}_m$ will stand for the ring of integers modulo $m$.

**Proposition 7.2** For a prime $p$, let the RS-admissible quadruple $(\ell, n, d, q=p^h)$ satisfy
\[
\left(1 - \rho_2 = \right) 1 - \frac{2}{\ell(\ell+1)} \leq d/n < 1.
\]
Assume in addition that either

(a) $n-d \mid q-1$ and the additive group of $\mathbb{Z}_{(q-1)/(n-d)}$ contains a weak Sidon set of size $\ell+1$, or

(b) $n-d = p^b$ for some integer $b$ and $\mathbb{Z}_{p^{b-h}}$ contains a weak Sidon set of size $\ell+1$.

Then,
\[
\Delta^\text{RS}_\ell(n, d; q) = \left\lfloor \tau_\ell(n, d) \right\rfloor - 1 = \left\lfloor \ell n/(\ell+1) - \ell(n-d)/2 \right\rfloor - 1.
\]

It is known that the additive group of $\mathbb{Z}_{(q-1)/(n-d)}$ contains a weak Sidon set of size $\ell+1$ whenever
\[
\ell^2 \cdot (1 + o(1)) < (q-1)/(n-d),
\]
where $o(1)$ stands for an expression that goes to zero as $\ell \to \infty$ [19, Theorem 1]. For the group in part (b) of Proposition 7.2, the known bounds imply a weak Sidon set of size $\ell$ whenever
\[
\ell^{2+o(1)} < (q-1)/(n-d),
\]
(see [3, Section 5]).

Observe that we have excluded the case $d = n$ (the repetition code) from Proposition 7.2. Here we have
\[
\Delta^\text{RS}_\ell(n, n; q) = \left\lfloor \tau_\ell(n, n) \right\rfloor - 1 = \left\lfloor \ell n/(\ell+1) \right\rfloor - 1
\]
only when $\ell < q$: there are $\ell+1$ codewords at Hamming distance $\leq \left\lfloor ((\ell n)/(\ell+1)) \right\rfloor$ from a word $v$ in which each of some $\ell+1$ elements of $\text{GF}(q)$ occurs at least $\lfloor n/(\ell+1) \rfloor$ times. When $\ell \geq q$ we obviously have $\Delta^\text{RS}_\ell(n, n; q) = n$.

Next we turn to the intermediate sub-intervals in (1.11), i.e., the range
\[
\frac{2}{\ell+1} < \frac{d}{n} < 1 - \frac{2}{\ell(\ell+1)};
\]
which is nonempty for $\ell \geq 3$. Propositions 7.3 and 7.4 identify cases where the GS bound is tight for code rate $(k-1)/n$ around 0.3 and list sizes 3 or 4.
Proposition 7.3 Let \((3, 10m+p+\nu, 7m+\nu+1, q)\) be an RS-admissible quadruple, where \(q = p^k\) for a prime \(p\); the integer pair \((\nu, \kappa)\) belongs to the set \(\{(\nu, \kappa) : -1 \leq \nu \leq 1, 1 \leq \kappa \leq 5-3\nu\}\); and \(m\) is a positive integer such either \((a)\) \(m \mid q-1\) and \(q \geq 11m\), or \((b)\) \(m \mid q\) and \(p \notin \{3, 5, 7\}\). Then,

\[
\Delta_3^{RS}(10m+\nu+\kappa, 7m+\nu+1; q) = [\tau_3(10m+\nu+\kappa, 7m+\nu+1)] - 1 = 4m + \nu .
\]

Proposition 7.4 Let \((4, 10m+\nu+\kappa, 7m+\nu+1, q)\) be an RS-admissible quadruple, where \((\nu, \kappa)\) is an integer pair in the set \(\{(2,1)\} \cup \{(\nu, \kappa) : -1 \leq \nu \leq 1, 1 \leq \kappa \leq 9-6\nu\}\) and \(q\) and \(m\) are powers of 2. Then,

\[
\Delta_4^{RS}(10m+\nu+\kappa, 7m+\nu+1; q) = [\tau_4(10m+\nu+\kappa, 7m+\nu+1)] - 1 = 4m + \nu .
\]

Propositions 7.5 and 7.6 below describe two cases where failing lists cannot be realized in RS codes though the respective \((\ell, r)\)-configuration (see Proposition 6.3) may be realized using non-algebraic codes. We conclude that \(\Delta_\ell^{RS}(n, d; q)\) exceeds the GS bound in these two cases.

Proposition 7.5 Let \(q \geq 11\) be a power of an odd prime. Then,

\[
\Delta_4^{RS}(10, 7; q) \geq \tau_4(10, 7) = 4 .
\]

In contrast, note that from Proposition 7.4 we have \(\Delta_4^{RS}(10, 7; q) = \tau_4(10, 7) - 1 = 3\) when \(q\) is even. Moreover, it follows from Theorem 4 in [23] that the list-4 decoding radius of the worst \([10, 4, 7]\) MDS code is 3 for every large enough (either even or odd) field size \(q\).

Proposition 7.6 For every \(h \geq 4\),

\[
\Delta_6^{RS}(11, 9; 2^h) \geq \tau_6(11, 9) = 6 .
\]

In Section 7.2, tools are developed for synthesizing and analyzing failing lists in RS codes. Section 7.3 contains the proofs of Propositions 7.1–7.6.

### 7.2 Realizing failing lists in RS codes

Throughout this section we fix the finite field \(F = GF(q)\), the list size \(\ell\), and an \([n, k, d]\) RS code \(C\) over \(F\) with a set of code locators \(\{\alpha_1, \alpha_2, \ldots, \alpha_n\} \subseteq F\). We let \(r\) be the unique integer such that \(1 - \delta = (k-1)/n \in [\rho_\ell, \rho_{\ell+1})\). We use the notation \([n]\) for the set \(\{1, 2, \ldots, n\}\).

Suppose that \(C\) contains a set \(C = \{c_0, c_1, \ldots, c_\ell\}\) which is an \((\ell, r)\)-configuration with respect to some partition vector \(P\) for which (6.5) and (6.6) are satisfied. Without loss of generality, assume that \(c_0\) is the zero codeword (otherwise, subtract \(c_0\) from each \(c_s\) to obtain another \((\ell, r)\)-configuration with respect to \(P\)). For every two indexes \(s, t\) such that \(0 \leq s < t \leq \ell\), the difference \(c_t - c_s\) is a codeword that is obtained by evaluating a polynomial of degree \(\leq k-1\) at the code locators. We denote this polynomial by \(a_{s,t} \cdot f_{s,t}(x)\), where \(a_{s,t} \in F \setminus \{0\}\) and \(f_{s,t}(x)\) is a monic polynomial of degree \(\leq k-1\).
For every subset $S_i \subseteq \{0, 1, \ldots, \ell\}$ of size $r$ and every subset $S_j' \subseteq S_j$ of size $r+1$, define

$$A_{S_i}(x) = \prod_{\mu \in I_i} (x - \alpha_\mu) \quad \text{and} \quad A_{S_j'}(x) = \prod_{\mu \in I_j'} (x - \alpha_\mu) . \quad (7.3)$$

By the definition of an $(\ell, r)$-configuration with respect to a partition vector, it follows that the polynomials $f_{s,t}(x)$ are given by

$$f_{s,t}(x) = \prod_{i : \{s,t\} \subseteq S_i} A_{S_i}(x) \cdot \prod_{j : \{s,t\} \subseteq S_j'} A_{S'_j}(x) \quad (7.4)$$

(the product over an empty set is defined as 1). A necessary condition on the existence of the configuration $\mathcal{L}$ in $\mathcal{C}$ is:

$$a_{s,t} \cdot f_{s,t}(x) = a_{0,s} \cdot f_{0,s}(x) - a_{0,t} \cdot f_{0,t}(x), \quad 0 < s < t \leq \ell . \quad (7.5)$$

Conversely, suppose that $\mathcal{P} = (I_i)_{i \in [n]}$ is a partition vector that satisfies (6.5) and (6.6), and let the polynomials $f_{s,t}(x)$ be defined by (7.4). If there are nonzero constants $a_{s,t} \in F$ that satisfy (7.5), then an $(\ell, r)$-configuration with respect to $\mathcal{P}$ exists. Based on Lemma 6.1 and Proposition 6.3, the following lemma is obtained.

**Lemma 7.7** Let $(I_i)_{i \in [n]}$ be a partition vector of $[n]$ that satisfies (6.5) and (6.6) and let the polynomials $f_{s,t}(x)$, $0 \leq s < t \leq \ell$, be defined by (7.4).

(a) If there are $\binom{\ell+1}{2}$ nonzero constants $a_{s,t} \in F$ such that (7.5) holds, then $\mathcal{C}$ contains a failing list that consists of the zero codeword and the following $\ell$ codewords

$$(f(a_1) f(a_2) \ldots f(a_\ell)), \quad f(x) \in \{a_{0,s} \cdot f_{0,s}(x)\}_{s=1}^\ell .$$

(b) In cases where (1.14) holds, the sufficient conditions in part (a) for the existence of a failing list of size $\ell+1$ are also necessary, and each polynomial $f_{s,t}(x)$ has degree $k-1$.

### 7.2.1 The difference condition and simple sets of polynomials

Three monic polynomials $f(x) = \sum_i f_i x^i$, $g(x) = \sum_i g_i x^i$, and $h(x) = \sum_i h_i x^i$ are said to satisfy the **difference condition** if there are $\tilde{f}, \tilde{g}, \tilde{h} \in F \setminus \{0\}$ for which

$$\tilde{h} \cdot h(x) = \tilde{f} \cdot f(x) - \tilde{g} \cdot g(x) . \quad (7.6)$$

Observe that every three polynomials in (7.5) satisfy the difference conditions.

**Lemma 7.8** Three distinct monic polynomials of the same degree $e$, $f(x) = \sum_{i \leq e} f_i x^i$, $g(x) = \sum_{i \leq e} g_i x^i$, and $h(x) = \sum_{i \leq e} h_i x^i$, satisfy the difference condition if and only if

$$(f_i - h_i)(f_j - g_j) = (f_i - g_i)(f_j - h_j) \quad \text{for every} \quad 0 \leq i, j \leq e . \quad (7.7)$$
Proof: The difference condition is satisfied if and only if there are \( \tilde{f}, \tilde{g}, \tilde{h} \in F \setminus \{0\} \) for which
\[
\tilde{h} \cdot h_i = \tilde{f} \cdot f_i - \tilde{g} \cdot g_i, \quad 0 \leq i \leq e, \tag{7.8}
\]
and since \( f(x), g(x), \) and \( h(x) \) are monic polynomials of the same degree \( e \), we obtain, in particular, that
\[
\tilde{h} = \tilde{f} - \tilde{g}. \tag{7.9}
\]
A nontrivial solution for \( \tilde{f}, \tilde{g}, \tilde{h} \) exists if and only if the following matrix is singular for every \( 0 \leq i \leq j \leq e \):
\[
\begin{pmatrix}
1 & -1 & -1 \\
f_i & -g_i & -h_i \\
f_j & -g_j & -h_j
\end{pmatrix}.
\]
This matrix, in turn, is singular if and only if (7.7) holds.

Now, the values of \( \tilde{f}, \tilde{g}, \tilde{h} \) must be all nonzero in every nontrivial solution of (7.6), as required by the difference condition: by (7.9), it is impossible that exactly two of them are zero, and if only one is zero, then, by combining (7.8) and (7.9) we obtain that two out of three polynomials \( f(x), g(x), \) and \( h(x) \) are identical; this, however, contradicts our assumption that these polynomials are distinct.

Our constructions that realize (7.4) and (7.5) will have the special structure defined below. A set of polynomials of degree \( e \) over \( F \) is simple over a set \( U \subseteq F \) if the following three conditions hold:

(S1) Each polynomial has \( e \) simple roots in \( U \).

(S2) Every two distinct polynomials in the set are relatively prime.

(S3) The polynomials differ only in the \( i \)th coefficient, for some \( i \). For example, they differ only in their constant term.

By Lemma 7.8, every three polynomials in a simple set satisfy the difference condition.

### 7.2.2 Rates above \( 2/(\ell (\ell+1)) \)

Lemma 7.9 below provides a sufficient condition on the existence of failing lists of size \( \ell+1 \) in \( C \). The statement of the lemma makes use of the following notation. Let \( \mathcal{P} = (I_i)_{i \in [\ell]} \) be a partition vector of \([n]\) and let \( f_{s,t}(x), 0 \leq s < t \leq \ell, \) be defined by (7.4). For every \( s, t, u \) such that \( 0 \leq s < t < u \leq \ell \), define
\[
g_{s,t,u}(x) = \gcd(f_{s,t}(x), f_{s,u}(x), f_{t,u}(x)) = \prod_{i : \{s,t,u\} \subseteq S_i} A_{S_i}(x) \cdot \prod_{j : \{s,t,u\} \subseteq S_j} A_{S_j}(x). \tag{7.10}
\]

**Lemma 7.9** Let \( \mathcal{P} = (I_i)_{i \in [\ell]} \) be a partition vector of \([n]\) for which (6.5) and (6.6) hold, and let \( f_{s,t}(x) \) and \( g_{s,t,u}(x) \) be the polynomials defined by (7.4) and (7.10), respectively. A failing list of size \( \ell+1 \) is contained in \( C \) if the following two conditions hold:

- For every \( 0 \leq s < t \leq \ell, \) the polynomials \( f_{0,s}(x), f_{0,t}(x), \) and \( f_{s,t}(x) \) satisfy the difference condition, and
• \( g_{s,t,u}(x) \) does not divide \( f_{0,s}(x) \) for every \( 0 < s < t < u \leq \ell \).

**Proof:** We show that the sufficient conditions of Lemma 7.7(a) hold. If \( f_{0,s}(x), f_{0,t}(x), \) and \( f_{s,t}(x) \) satisfy the difference condition, then, by definition, there must be nonzero \( a_{0,s}, a_{0,t}, a_{s,t} \in F \) such that

\[
a_{s,t} \cdot f_{s,t}(x) = a_{0,s} \cdot f_{0,s}(x) - a_{0,t} \cdot f_{0,t}(x) .
\]

(7.11)

In case where \( \ell = 2 \), we are done.

Turning to larger values of \( \ell \), we need to show that \( f_{0,s}(x) \) has the same coefficient \( a_{0,s} \) everywhere in (7.5), independently of \( t \). Given \( s \in [\ell-2] \), consider any indexes \( t \) and \( u \) such that \( s < t < u \leq \ell \). There must be nonzero \( a_{0,s}, a_{0,t}, a_{s,t} \in F \) satisfying (7.11) and, by the same arguments, there must be nonzero \( a_{0,u}, a_{t,u}, a_{s,u}, a'_{0,s} \in F \) such that

\[
a_{t,u} \cdot f_{t,u}(x) = a_{0,t} \cdot f_{0,t}(x) - a_{0,u} \cdot f_{0,u}(x) \quad (7.12)
\]

\[
a_{s,u} \cdot f_{s,u}(x) = a'_{0,s} \cdot f_{0,s}(x) - a_{0,u} \cdot f_{0,u}(x) . \quad (7.13)
\]

Subtracting (7.13) from the sum of (7.11) and (7.12) results in

\[
a_{s,t} \cdot f_{s,t}(x) + a_{t,u} \cdot f_{t,u}(x) - a_{s,u} \cdot f_{s,u}(x) = (a_{0,s} - a'_{0,s}) \cdot f_{0,s}(x) . \quad (7.14)
\]

Clearly, \( g_{t,u}(x) \) divides the left-hand side of (7.14). However, according to the assumptions of the lemma, \( g_{t,u}(x) \) does not divide \( f_{0,s}(x) \). We therefore conclude that \( a'_{0,s} \neq a_{0,s} \), i.e., \( f_{0,s}(x) \) indeed has the same coefficient \( a_{0,s} \) everywhere in (7.5), independently of \( t \).

Corollaries 7.10 and 7.11 below are derived from Lemma 7.9 and are used in Section 7.3 to indicate families of RS codes that contain failing lists of size \( \ell+1 \). Corollary 7.10 covers only the high-rate range \( (k-1)/n \geq 1 - (2/(\ell+1))(= \rho_k) \), while Corollary 7.11 applies to \( (k-1)/n > 2/(\ell(\ell+1)) (= \rho_2) \).

**Corollary 7.10** Let the triple \((\ell, n, k)\) of positive integers be such that \( (k-1)/n \geq 1 - (2/(\ell+1)) \). A failing list is contained in \( C \) if there is some partition vector \( \mathcal{P} = (I_1, I_2, \ldots, I_{\ell+1}, I'_1) \) of \([n]\) such that the following holds:

(a) (6.5) and (6.6) are satisfied, with equality in (6.5), and

(b) for every \( 0 \leq s < t \leq \ell \) in (7.4), the respective polynomials \( f_{0,s}(x), f_{0,t}(x), \) and \( f_{s,t}(x) \), each of degree \( k-1 \), are distinct and satisfy the difference condition.

**Proof:** We show that whenever \( r = \ell > 2 \), for every \( 0 < s < t < u \leq \ell \) the polynomial \( g_{s,t,u}(x) \) in (7.10) does not divide \( f_{0,s}(x) \). The existence of a failing list will then follow from Lemma 7.9.

Without loss of generality we can assume that the sets \( S_i \) are defined so that \( S_1 = \{1, 2, \ldots, \ell\} \). For every \( 0 < s < t < u \), the polynomial \( g_{s,t,u}(x) \) does not divide \( f_{0,s}(x) \) if and only if \( \deg A_{S_1}(x) > 0 \). Assume to the contrary that \( \deg A_{S_1}(x) = 0 \). Since \( A_{S_1}(x) = f_{s,t}(x)/g_{0,s,t}(x) \), it then follows that \( \deg g_{0,s,t}(x) = k-1 \); therefore, \( f_{0,s}(x) = g_{0,s,t}(x) = f_{0,t}(x) \), contradicting our assumption that \( f_{0,s}(x) \) and \( f_{0,t}(x) \) are distinct. ■
Corollary 7.11 Let the triple \((\ell, n, k)\) of positive integers be such that \((k-1)/n > 2/(\ell(\ell+1))\), and let \(r > 1\), \(\gamma > 0\), and \(\gamma' \geq 0\) be integers for which (6.7) holds. Let \(P = (I_1)|| (I'_j)_i\) be a partition vector of \([n]\) in which \(|I_1| = \gamma\) and \(|I'_j| = \gamma'\), and let \(f_s,t(x)\) and \(g_s,t,u(x)\) be the polynomials defined by (7.4) and (7.10), respectively. Suppose that for every \(0 \leq s < t \leq \ell\), there is a polynomial divisor \(\lambda_{s,t}(x)\) of \(g_{s,t,u}(x)\) for which the set \(\{f_{0,s}(x)/\lambda_{s,t}(x), f_{0,t}(x)/\lambda_{s,t}(x), f_{s,t}(x)/\lambda_{s,t}(x)\}\) is simple over the set of code locators of \(C\). Then \(C\) contains a failing list of size \(\ell+1\).

Proof: By Lemma 7.9, it suffices to show that \(g_{s,t,u}(x)\) does not divide \(f_{0,s}(x)\) for every \(0 < s < t < u \leq \ell\). If \(2 < r \leq \ell\), there is a nonempty partition element \(I_1\) in \(P\) that corresponds to a subset \(S_1 \subseteq \{0,1, \ldots, \ell\}\) such that \(\{s,t,u\} \subseteq S_1\) while \(0 \notin S_1\). In this case, \(A_{S_1}(x)\) divides \(g_{s,t,u}(x)\) yet it does not divide \(f_{0,s}(x)\); therefore, \(g_{s,t,u}(x)\) does not divide \(f_{0,s}(x)\), as required.

Assume now that \(2 = r < \ell\). Since \((k-1)/n > 2/(\ell(\ell+1))\), there must exist a nonempty partition element \(I'_j\) in \(P\) that corresponds to a subset \(S'_j = \{s,t,u\}\) of \(\{0,1, \ldots, \ell\}\). Since \(|I'_j| = \gamma' > 0\), the polynomial \(A_{S'_j}(x)\) divides \(g_{s,t,u}(x)\) but not \(f_{0,s}(x)\); thus, \(g_{s,t,u}(x)\) does not divide \(f_{0,s}(x)\), as required.

7.2.3 The low-rate range: \(0 < (k-1)/n \leq 2/(\ell(\ell+1))\)

Suppose that the rate of \(C\) satisfies \((k-1)/n < \rho_2 = 2/(\ell(\ell+1))\) and that \(C\) contains an \((\ell,1)\)-configuration \(L\). At most two out of the \(\ell+1\) words in \(C\) agree on every position and, so, (7.4) becomes \(f_{s,t}(x) = f_{s,t}(x)/g_{0,s,t}(x) = A_{S'_j}(x)\) for \(S'_j = \{s,t\}\). Suppose now that \((k-1)/n = \rho_2\) and that \(C\) is an \((\ell,2)\)-configuration; here, \(f_{s,t}(x) = f_{s,t}(x)/g_{0,s,t}(x) = A_{S_1}(x)\) for \(S_1 = \{s,t\}\). In both cases, the set \(\{f_{s,t}(x)\}_{s,t}\) already satisfies conditions (S1) and (S2) for being simple over the set of code locators of \(C\). However, it turns out that when \(\ell > 2\) in any of those two cases, taking the set \(\{f_{s,t}(x)\}_{s,t}\) to be simple over the set of code locators does not guarantee the existence of multipliers \(\{a_{s,t}\}_{s,t}\) for which (7.5) holds. An auxiliary condition on the coefficients of \(\{f_{s,t}(x)\}_{s,t}\) is needed in this case, as stated in the following lemma.

Lemma 7.12 Let the triple \((\ell, n, k)\) of positive integers be such that \((k-1)/n \leq 2/(\ell(\ell+1))\). Suppose there exists a set \(\{f_{0,1}(x), f_{0,2}(x), \ldots, f_{0,1}(x)\}\) of \(\binom{\ell+1}{2}\) distinct polynomials of degree \(k-1\) over \(F\) that is simple over a subset \(U\) of size \(\binom{\ell+1}{2}(k-1)\) of the set of code locators of \(C\). Let \(e\) be the \((unique)\) coefficient index in which the polynomials differ, and denote by \(\psi_{s,t}\) the \(e\)th coefficient of \(f_{s,t}(x)\). Assume that when \(\ell > 2\), the coefficients \(\psi_{s,t}\) satisfy the \(\binom{\ell-1}{2}\) equations

\[
(\psi_{1,s}-\psi_{0,1})(\psi_{1,t}-\psi_{0,t})(\psi_{s,t}-\psi_{0,s}) = (\psi_{1,s}-\psi_{0,s})(\psi_{1,t}-\psi_{0,t})(\psi_{s,t}-\psi_{0,t}), \quad 1 < s < t \leq \ell .
\]

Then there exist nonzero \(a_{0,1}, a_{0,2}, \ldots, a_{0,t} \in F\) such that the zero word and the following \(\ell\) words,

\[
(f(a_1) f(a_2) \ldots f(a_n)), \quad f(x) \in \{a_{0,s} \cdot f_{0,s}(x)\}_{s=1}^\ell ,
\]

form a failing list in \(C\).
Proof: Our proof is based on Lemma 7.7(a). To this end, we first find a partition vector \( \mathcal{P} = (I_i)_{i} \parallel (I'_j)_j \) of \([n]\) that satisfies (6.5) and (6.6) and that allows us to express the polynomials \( f^*_{s,t}(x) \) in the form (7.4). When \((k-1)/n = \rho_2\) we select \( \mathcal{P} \) to be proper and for every \( S_i = \{s,t\} \) we let \( I_i = \{\mu : f^*_{s,t}(\alpha_\mu) = 0\} \).

When \((k-1)/n < \rho_2\), we select \( \mathcal{P} = (I_i)_{i} \parallel (I'_j)_j \) so that for \( S'_j = \{s,t\} \) the partition element \( I'_j \) is given by \( \{\mu : f^*_{s,t}(\alpha_\mu) = 0\} \). Each of the \( \ell + 1 \) partition elements \( I_i \), which correspond to singleton subsets \( S_i \), contains at least \([n-|U|]/(\ell+1)\) of the remaining elements of \([n]\). Since the various polynomials \( f^*_{s,t}(x) \) are all distinct, \( \mathcal{P} \) is indeed a partition vector. It is also clear that \( \mathcal{P} \) satisfies (6.5) with equality. As for (6.6), for every \( s = 0,1,\ldots,\ell \),

\[
\sum_{i:s \in S_i} |I_i| + \sum_{j:s \in S'_j} |I'_j| \geq \ell(k-1) + \left\lfloor \frac{n-|U|}{\ell+1} \right\rfloor = n - \left\lfloor \frac{m}{\ell+1} \right\rfloor = n - [\tau_\ell(n,k)].
\]

Given the partition vector \( \mathcal{P} \), we have \( f^*_{s,t}(x) = f_{s,t}(x) \), where \( f_{s,t}(x) \) are given by (7.4). By Lemma 7.7(a), all we still need to show is that there are nonzero coefficients \( a_{s,t} \), \( 0 \leq s < t \leq \ell \), for which (7.5) holds. We distinguish between three cases, according to the value of \( \ell \) (omitting the obvious case \( \ell = 1 \)).

Case 1: \( \ell = 2 \). The three polynomials \( f_{0,1}(x) \), \( f_{0,2}(x) \), and \( f_{1,2}(x) \) satisfy condition (S3) of a simple set; therefore, by Lemma 7.8, they satisfy the difference condition.

Case 2: \( \ell = 3 \). Since \( f_{0,1}(x) \), \( f_{0,2}(x) \), \( f_{1,2}(x) \) satisfy condition (S3), the set of linear equations (7.5) has a nontrivial solution for the unknowns \( a_{0,1},a_{0,2},\ldots,a_{2,3} \) if and only if there is a nontrivial solution for the following set of equations:

\[
\begin{pmatrix}
1 & -1 & 0 & -1 & 0 & 0 \\
0 & 1 & 0 & -1 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 \\
\psi_{0,1} & -\psi_{0,2} & 0 & -\psi_{1,2} & 0 & 0 \\
\psi_{0,1} & 0 & -\psi_{0,3} & 0 & -\psi_{1,3} & 0 \\
0 & \psi_{0,2} & -\psi_{0,3} & 0 & 0 & -\psi_{2,3}
\end{pmatrix}
\begin{pmatrix}
a_{0,1} \\
a_{0,2} \\
a_{0,3} \\
a_{1,2} \\
a_{1,3} \\
a_{2,3}
\end{pmatrix}
= 0. \tag{7.16}
\]

However, the determinant of the matrix in (7.16) is zero if and only if

\[
(\psi_{1,2} - \psi_{0,1})(\psi_{1,3} - \psi_{0,3})(\psi_{2,3} - \psi_{0,2}) = (\psi_{1,2} - \psi_{0,2})(\psi_{1,3} - \psi_{0,1})(\psi_{2,3} - \psi_{0,3});
\]

this is condition (7.15) for \( \ell = 3 \). Furthermore, if one of the elements \( a_{0,1},a_{0,2},\ldots,a_{2,3} \) is zero, then either all these elements are zero, or else \( \psi_{s',t'} = \psi_{s'',t''} \) for some \( (s',t') \neq (s'',t'') \), where \( 0 \leq s' < t' \leq 3 \) and \( 0 \leq s'' < t'' \leq 3 \); yet, the latter contradicts our assumption that \( f_{0,1}(x),f_{0,2}(x),\ldots,f_{2,3}(x) \) are all distinct. Therefore, in a nontrivial solution for \( a_{0,1},a_{0,2},\ldots,a_{2,3} \), all these elements are nonzero.

Case 3: \( \ell > 3 \). Fix some \( s \) in the range \( 1 < s \leq \ell-2 \), and consider another index \( t \) in the range \( s < t \leq \ell-1 \). Following the analysis of Case 2, there must exist nonzero \( a_{0,1},a_{0,s},a_{0,t},a_{1,s},a_{1,t},a_{s,t} \in F \) such that

\[
\begin{align*}
as_{s,t} \cdot f_{s,t}(x) &= a_{0,s} \cdot f_{0,s}(x) - a_{0,t} \cdot f_{0,t}(x) \\
an_{1,s} \cdot f_{1,s}(x) &= a_{0,1} \cdot f_{0,1}(x) - a_{0,s} \cdot f_{0,s}(x) \\
an_{1,t} \cdot f_{1,t}(x) &= a_{0,1} \cdot f_{0,1}(x) - a_{0,t} \cdot f_{0,t}(x). \tag{7.17}
\end{align*}
\]
Let \( u \) be in the range \( t < u \leq \ell \); there are five nonzero coefficients \( a_{0,s}, a'_{1,s}, a_{0,u}, a_{1,u}, a_{s,u} \) such that

\[
\begin{align*}
a_{s,u} \cdot f_{s,u}(x) &= a'_{0,s} \cdot f_{0,s}(x) - a_{0,u} \cdot f_{0,u}(x) \\
a'_{1,s} \cdot f_{1,s}(x) &= a_{0,1} \cdot f_{0,1}(x) - a'_{0,s} \cdot f_{0,s}(x) \\
a_{1,u} \cdot f_{1,u}(x) &= a_{0,1} \cdot f_{0,1}(x) - a_{0,u} \cdot f_{0,u}(x). \\
\end{align*}
\]

Combining (7.17) and (7.18) results in

\[
(a_{1,s} - a'_{1,s}) \cdot f_{1,s}(x) = (a'_{0,s} - a_{0,s}) \cdot f_{0,s}(x),
\]

and since \( f_{1,s}(x) \) and \( f_{0,s}(x) \) are relatively prime, it follows that \( a'_{0,s} = a_{0,s} \) and \( a'_{1,s} = a_{1,s} \). Hence, the same nonzero constant \( a_{0,s} \) multiplies \( f_{0,s}(x) \) in (7.5), independently of \( t \).

### 7.3 Proof of main results for RS codes

In this section, we prove Propositions 7.1–7.6 using the tools developed in Section 7.2.

#### 7.3.1 The function \( \Delta_{\ell}^{\text{RS}}(n, d; q) \)

The following simple relations satisfied by \( \Delta_{\ell}^{\text{RS}}(n, d; q) \) are used in some of the proofs given in this section.

**Proposition 7.13** Let \((\ell, n, d, q)\) be an RS-admissible quadruple. Then,

(a) \( \Delta_{\ell}^{\text{RS}}(n-1, d-1) \leq \Delta_{\ell}^{\text{RS}}(n, d; q) \leq \Delta_{\ell}^{\text{RS}}(n-1, d-1; q) + 1 \) for \( d > 1 \).

(b) \( \Delta_{\ell}^{\text{RS}}(n, d; q) \leq \Delta_{\ell}^{\text{RS}}(n-1, d; q) \) for \( d < n \).

**Proof:** Part (a): Let \( \mathcal{C} \) be an \([n, k, d]\) RS code over \( \text{GF}(q) \) where \( k < n \) (\( d > 1 \)) and let \( \mathcal{C}' \) be obtained by deleting the last coordinate from each codeword of \( \mathcal{C} \). A list-\( \ell \) decoder for \( \mathcal{C} \) can be obtained by truncating the last coordinate from the received word and applying a list-\( \ell \) decoder for \( \mathcal{C}' \) to the resulting word. Hence, \( \Delta_{\ell}(\mathcal{C}) \geq \Delta_{\ell}(\mathcal{C}') \), and, so, \( \Delta_{\ell}(n, d; q) \geq \Delta_{\ell}(n-1, d-1; q) \). On the other hand, a list-\( \ell \) decoder for \( \mathcal{C}' \) can be obtained by appending an arbitrary \( n \)-th coordinate to the received word, followed by an application of a list-\( \ell \) decoder for \( \mathcal{C} \). Therefore, \( \Delta_{\ell}(\mathcal{C}') \geq \Delta_{\ell}(\mathcal{C}) - 1 \) and, since \( \mathcal{C}' \) can be any \([n-1, k, d-1]\) RS code, \( \Delta_{\ell}(n-1, d-1; q) \geq \Delta_{\ell}(n, d; q) - 1 \).

Part (b): Every \([n-1, k-1, d]\) RS code \( \mathcal{C} \) over \( \text{GF}(q) \) with \( n \leq q \) can be extended to an \([n, k, d]\) (generalized) RS code \( \overline{\mathcal{C}} \) over \( \text{GF}(q) \) by adding one column to the parity-check matrix of \( \mathcal{C} \); (see [27, Section 10.8]). Therefore, a list-\( \ell \) decoder for \( \mathcal{C} \) can be obtained by appending a zero coordinate to the received word and then applying a list-\( \ell \) decoder for \( \overline{\mathcal{C}} \). Hence, \( \Delta_{\ell}(\mathcal{C}) \geq \Delta_{\ell}(\overline{\mathcal{C}}) \) and, so, \( \Delta_{\ell}(n-1, d; q) \geq \Delta_{\ell}(n, d; q) \).

#### 7.3.2 Types of simple sets of polynomials

In some of our proofs, we will use two types of sets of polynomials that are simple over certain sets \( U \), as follows.
**Type 1:** Assume that $e \mid q-1$ and let $\alpha$ be a primitive element in $F = \text{GF}(q)$. The set
\[
\left\{ x^e - \alpha^i : 0 \leq i < (q-1)/e \right\}
\]
is simple over $F \setminus \{0\}$.

**Type 2:** Assume that $q = p^h$ and $e = p^b < q$. If we regard $F = \text{GF}(q)$ as a linear space over $\text{GF}(p)$, then the $p^h$ elements of every $b$-dimensional subspace $F'$ of $F$ are the roots of some nonzero linearized polynomial $\eta(x) = \sum_{i=0}^{b} \eta_i x^{p^i}$ over $F$ (see [26, Ch. 4] or [27, Ch. 4]). The polynomial $\eta(x)$ defines a linear mapping $\eta : F \to F$ over $\text{GF}(p)$. The range $R_\eta$ of the mapping $x \mapsto \eta(x)$ over $F$ is a subspace of $F$ of dimension $h-b$ in which every two distinct elements have disjoint sets of $p^b$ pre-images under $\eta$. The set
\[
\left\{ \eta(x) - \beta : \beta \in R_\eta \right\}
\]
is thus simple over $F$.

### 7.3.3 The high-rate range: Proposition 7.1

**Proof of Proposition 7.1:** We consider here codes at rates $(k-1)/n \geq 1 - (2/(\ell+1)) = \rho_k$. Starting with the case $k = n$, we have $\ell < 2n \leq 1 + (q-1)n$ for all RS-admissible quadruples $(\ell, n, d, q) \neq (3, 2, 1, 2)$; so, in this case, $\Delta_1(n, 1; q) = 0$.

We assume from now on that $d = n - k + 1$ is an even number (in such cases (1.14) holds); the case of odd $d$ follows from Proposition 7.13(a). We show that there is an $[n, k, d]$ RS code $C$ over $F = \text{GF}(q)$ that contains a failing list of size $\ell+1$.

Let $S'_1$ be the set $\{0, 1, \ldots, \ell\}$ and $S_1, S_2, \ldots, S_{\ell+1}$ be the subsets of $S'_1$ of size $\ell$. Using any of the constructions of simple sets in Section 7.3.2, we let $\{A_{S_i}(x)\}_{i=1}^{\ell+1}$ be a simple set over $F$, where $\deg A_{S_i}(x) = d/2$ for every $i$. We denote by $U_j$ the set of $d/2$ roots of $A_{S_j}(x)$ in $F$ and by $U$ the union $\bigcup_{j=1}^{\ell+1} U_i$. Also, define $U'_1$ to be a subset of $F \setminus U$ of size $n - (\ell+1)d/2$ and let
\[
A_{S_i}(x) = \prod_{\alpha \in U'_1} (x - \alpha) .
\]
Define $P$ to be a partition vector $(I_1, I_2, \ldots, I_{\ell+1}, I'_1)$ of $[n]$ with $|I_i| = d/2$ and $|I'_1| = n - (\ell+1)d/2$, and let $C$ be defined by the code locators $\alpha_1, \alpha_2, \ldots, \alpha_n$, where $U_i = \{\alpha_{\mu}\}_{\mu \in I_i}$ and $U'_1 = \{\alpha_{\mu}\}_{\mu \in I'_1}$.

By construction, $P$ satisfies both (6.5) and (6.6) with equality. Finally, let the polynomials $f_{i,t}(x)$ be defined by (7.4). For every $s < t$, the set $\left\{ f_{S_1}(x), f_{S_2}(x), f_{S_3}(x) \right\}$ contains three different polynomials from $\{A_{S_i}(x)\}_{i=1}^{\ell+1}$ and is therefore a simple set over $U$. Hence, the polynomials $f_{0,s}(x)$, $f_{0,t}(x)$, and $f_{i,t}(x)$ satisfy the difference condition. Corollaries 7.10 and 7.11 now imply that $C$ contains a failing list of size $\ell+1$.

### 7.3.4 The mid-rate range: Propositions 7.3–7.6

**List-3 decoders for RS codes at rates $\approx 0.3$**

**Proof of Proposition 7.3:** It suffices to prove the proposition for $(\nu, \kappa) = (-1, 1)$, in which case $\tau_3(10m, 7m) = 4m$; the results for the remaining values of $(\nu, \kappa)$ follow from
Proposition 7.13: for $\kappa = 1$, the result follows from Part (a) of Proposition 7.13, and then, for every fixed $\nu$, it follows from Part (b). We construct an $[n=10m, k=3m+1, d=7m]$ RS code $C$ over $F$ that contains a failing list of size $\ell+1 = 4$; note that here $r = 2$ and that $\gamma = \gamma' = m$ satisfy (6.7).

Part (a): We assume that the field size $q$ is such that $q-1 = m \cdot b$ where $b \geq 11$, and we let $\alpha$ be an element of order $b$ in the multiplicative group of $F = \text{GF}(q)$. We define six polynomials $A_{(s,t)}(x), 0 \leq s < t \leq 3$, and four polynomials $A_{(s,t,u)}(x), 0 \leq s < t < u \leq 3$ as follows:

\[
\begin{align*}
A_{(0,1)}(x) &= x^m - \alpha^2, \\
A_{(1,2)}(x) &= x^m - \alpha^3, \\
A_{(2,3)}(x) &= x^m - \alpha^9, \\
A_{(0,1,2)}(x) &= x^m - \alpha^{11}, \\
A_{(0,1,3)}(x) &= x^m - \alpha^5, \\
A_{(0,2,3)}(x) &= x^m - \alpha^6, \\
A_{(1,2,3)}(x) &= x^m - \alpha^4.
\end{align*}
\]

Note that any two of these ten polynomials are relatively prime, and each has $m$ simple roots in $F$.

For every $S_i = \{s, t\}$ (respectively, $S'_i = \{s, t, u\}$), let $U_i$ (respectively, $U'_i$) be the set of roots of $A_{S_i}(x)$ (respectively, $A_{S'_i}(x)$) in $F$. Define accordingly a partition vector $P = (I_i)_{i=1}^6$, $(I'_i)_{i=1}^3$ such that $U_i = \{\alpha^i\}_{i \in \mathbb{Z}}$ and $U'_i = \{\alpha^i\}_{i \in \mathbb{Z}}$. Denote by $U$ the union of $\bigcup_{i=1}^6 U_i$ and $\bigcup_{i=1}^3 U'_i$ and define $C$ to be the $[10m, 3m+1, 7m]$ RS code over $F$ whose set of code locators is $U$.

Next, we define the polynomials $f_{s,t}(x)$ by (7.4) and obtain

\[
\begin{align*}
f_{0,1}(x) &= (x^m - \alpha^2)(x^m - \alpha^5)(x^m - \alpha^{11}), \\
f_{0,2}(x) &= (x^m - \alpha)(x^m - \alpha^6)(x^m - \alpha^{11}), \\
f_{0,3}(x) &= (x^m - \alpha^5)(x^m - \alpha^6)(x^m - \alpha^7), \\
f_{1,2}(x) &= (x^m - \alpha^3)(x^m - \alpha^4)(x^m - \alpha^{11}), \\
f_{1,3}(x) &= (x^m - \alpha^4)(x^m - \alpha^5)(x^m - \alpha^7), \\
f_{2,3}(x) &= (x^m - \alpha^4)(x^m - \alpha^5)(x^m - \alpha^8).
\end{align*}
\]

Similarly, we define the polynomials $g_{s,t}(x)$ by (7.10), and the sets

\[
\begin{align*}
\{ f_{0,s,t}(x), g_{0,s,t}(x), f_{s,t}(x), g_{s,t}(x) \}, \quad 0 < s < t \leq 3,
\end{align*}
\]

are given by

\[
\begin{align*}
\{ f_{0,1}(x), f_{0,2}(x), f_{1,2}(x) \} &= \{ (x^m - \alpha^2)(x^m - \alpha^5), (x^m - \alpha)(x^m - \alpha^6), (x^m - \alpha^3)(x^m - \alpha^4) \}, \\
\{ f_{0,1}(x), f_{0,3}(x), f_{1,3}(x) \} &= \{ (x^m - \alpha^2)(x^m - \alpha^{11}), (x^m - \alpha^6)(x^m - \alpha^7), (x^m - \alpha^4)(x^m - \alpha^9) \}, \\
\{ f_{0,2}(x), f_{0,3}(x), f_{2,3}(x) \} &= \{ (x^m - \alpha^5)(x^m - \alpha^{11}), (x^m - \alpha^5)(x^m - \alpha^7), (x^m - \alpha^4)(x^m - \alpha^8) \}.
\end{align*}
\]

Each of these three sets of polynomials is simple over $U$, since the three polynomials in each set differ only in their coefficient of $x^m$. Applying Corollary 7.11 to the partition vector $(I_i)_{i=1}^6, (I'_i)_{i=1}^3$, it follows that $C$ contains a failing list of size 4.

Part (b): We assume that the field size $q$ is $p^b$ for a prime $p \geq 11$ and that $m = p^b$ for $b < h$; the case $p = 2$ is omitted, as it is covered by Proposition 7.4 (to be proved right below). Let $\eta(x)$ be a linearized polynomial of degree $m$ over $F = \text{GF}(q)$ that has $m$ simple roots in $\text{GF}(q)$. Let $\beta$ be a nonzero element in the range of the mapping $\eta : F \to F$; by linearity, the (distinct) elements $0, \beta, 2\beta, \ldots, 10\beta$ are also in that range. As in part (a), we define six
polynomials \( A_{(s,t)}(x) \), \( 0 \leq s < t \leq 3 \), and four polynomials \( A_{(s,t,u)}(x) \), \( 0 \leq s < t < u \leq 3 \), each having \( m \) simple roots in \( F \) and every two are relatively prime:

\[
\begin{align*}
A_{(0,1)}(x) &= \eta(x) - 2\beta, & A_{(0,2)}(x) &= \eta(x) - \beta, & A_{(0,3)}(x) &= \eta(x) - 7\beta, \\
A_{(1,2)}(x) &= \eta(x) - 3\beta, & A_{(1,3)}(x) &= \eta(x) - 9\beta, & A_{(2,3)}(x) &= \eta(x) - 8\beta, \\
A_{(0,1,2)}(x) &= \eta(x) - 11\beta, & A_{(0,1,3)}(x) &= \eta(x) - 5\beta, & A_{(0,2,3)}(x) &= \eta(x) - 6\beta, \\
A_{(1,2,3)}(x) &= \eta(x) - 4\beta.
\end{align*}
\]

The proof now continues as in part (a); in particular, the sets (7.19) that result in this case are simple, as the three polynomials in each set differ only in their constant term.

The failing list in Figure 6.1 is obtained from the construction in the proof of part (b) by taking \( F = \text{GF}(11) \), \( m = 1 \), \( \eta(x) = x \), and \( \beta = 1 \).

**List-4 decoders for RS codes at rates \( \approx 0.3 \)**

**Proof of Proposition 7.4:** We prove the proposition for \((\nu, \kappa) = (-1, 1)\), in which case \( \tau_1(10m, 7m) = 4m \); the results for the other values for \((\nu, \kappa)\) extend directly from Proposition 7.13. We construct an \([n=10m, k=3m+1, d=7m]\) RS code \( C \) over \( F = \text{GF}(2^h) \) that contains a failing list of size \( \ell+1 = 5 \); here \( r = 3 \), and the values \( \gamma = m \) and \( \gamma' = 0 \) satisfy (6.7).

Let \( \eta(x) \) be a linearized polynomial of degree \( m = 2^h \leq 2^{h-4} \) over \( F \) that has \( m \) simple roots in \( F \). The range, \( R_\eta \), of the mapping \( x \mapsto \eta(x) \) over \( F \) is a linear space of dimension \( h-b \geq 4 \) over \( \text{GF}(2) \); therefore, one can find four elements \( \beta_0, \beta_1, \beta_2, \beta_3 \in R_\eta \) that are linearly independent over \( \text{GF}(2) \). We represent each of the 16 elements \( \sum_{i=0}^3 \epsilon_i \beta_i \), \( \epsilon_i \in \text{GF}(2) \), as a 4-tuple \((\epsilon_0, \epsilon_1, \epsilon_2, \epsilon_3)\).

Define the ten polynomials \( A_{(s,t,u)}(x) \), \( 0 \leq s < t < u \leq \ell \), as follows:

\[
\begin{align*}
A_{(0,1,2)}(x) &= \eta(x) - (1000), & A_{(0,1,3)}(x) &= \eta(x) - (0100), \\
A_{(0,1,4)}(x) &= \eta(x) - (0010), & A_{(0,2,3)}(x) &= \eta(x) - (0001), \\
A_{(0,2,4)}(x) &= \eta(x) - (0111), & A_{(0,3,4)}(x) &= \eta(x) - (1011), \\
A_{(1,2,3)}(x) &= \eta(x) - (1001), & A_{(1,2,4)}(x) &= \eta(x) - (1111), \\
A_{(1,2,4)}(x) &= \eta(x) - (0011), & A_{(2,3,4)}(x) &= \eta(x) - (0110).
\end{align*}
\]

For every subset \( S_i \) of \( \{0, 1, 2, 3, 4\} \) of size \( 3 \), let \( U_i \) denote the set of the \( \gamma = m = 2^h \) roots of \( A_S(x) \) in \( F \), and denote by \( U \) the union \( \bigcup_{i=1}^{10} U_i \). Define the partition vector \( P = (I_i)_{i=1}^{10} \) so that \( U_i = \{\alpha_{i\mu}\}_{\mu \in I_i} \). The code \( C \) is now defined as a \([10m, 3m+1, 7m]\) RS code over \( F \) whose set of code locators is \( U \).

Let the polynomials \( f_{s,t}(x) \) and \( g_{s,t,u}(x) \) be defined by (7.4) and (7.10) respectively. It can be verified that each of the six sets

\[
\left\{ \frac{f_{0,1}(x)}{g_{0,1,2}(x)}, \frac{f_{0,2}(x)}{g_{0,1,3}(x)}, \frac{f_{0,3}(x)}{g_{0,1,4}(x)} \right\}, \quad 0 < s < t \leq 4,
\]

is simple over \( F \). In particular, the polynomials—each of degree \( 2m \)—in every set differ only in their constant terms. For example,

\[
\begin{align*}
\frac{f_{0,1}(x)}{g_{0,1,2}(x)} &= \ A_{(0,1,3)}(x) \cdot A_{(0,1,4)}(x) = x^{2m} + (0110) \cdot x^m + (0100) \cdot (0010), \\
\frac{f_{0,2}(x)}{g_{0,1,3}(x)} &= \ A_{(0,2,3)}(x) \cdot A_{(0,2,4)}(x) = x^{2m} + (0110) \cdot x^m + (0001) \cdot (0111), \\
\frac{f_{0,3}(x)}{g_{0,1,4}(x)} &= \ A_{(1,2,3)}(x) \cdot A_{(1,2,4)}(x) = x^{2m} + (0110) \cdot x^m + (1001) \cdot (1111)
\end{align*}
\]
The parameters defined by $\ell(n,d,q) = (4,10,7,q)$, where $q$ is odd. The next lemma characterizes the structure of a failing list of size 5 in a any $(10, M, 7)$ code over any field.

**Lemma 7.14** Let $q = p^h$ where $p$ is a prime. Every failing list of size 5 in a $(10, M, 7)$ code over GF($q$) is a $(4,3)$-configuration with respect to a proper partition vector $P = (I_i)_{i=1}^{10}$, where $|I_i| = 1$ for all $i$. Every failing list of size 5 thus corresponds to the complete BIBD(5,3,3).

**Proof:** The parameters $\ell = 4$, $r = 3$, $n = \left(\frac{\ell+1}{r}\right) = 10$, and $d = 7$ satisfy (1.14). Therefore, by Proposition 6.3, every failing list of size 5 forms an $(4,3)$-configuration with respect to a partition vector $P = (I_i)_{i=1}^{10}|| (P')_j = 1$ for which (6.5) and (6.6) hold with equality. Furthermore, since $\rho(r) = \rho(4) = 3/10 = 1 - d/n$, the partition vector $P$ is proper: exactly $r = 3$ codewords agree on every position. We next show that each set $I_i$ has size 1.

Assume to the contrary; since $\sum_{i=1}^{10} |I_i| = 10$, at least one of the partition elements, say $I_1$, is empty. Without loss of generality, let $S_1 = \{0, 1, 2\}$ and let the sets $I_2$ through $I_7$ correspond, respectively, to $S_2 = \{0, 1, 3\}$, $S_3 = \{0, 1, 4\}$, $S_4 = \{0, 2, 3\}$, $S_5 = \{0, 2, 4\}$, $S_6 = \{1, 2, 3\}$, and $S_7 = \{1, 2, 4\}$. We have,

$$\sum_{i=2}^{7} |I_i| = \sum_{0 \leq i < j \leq 2} \left( \sum_{i : \{s, t\} \subseteq S} |I_i| \right) = 9,$$

where the second equality follows from the equality in (6.5). Hence, either $|I_2| + |I_4| + |I_6| \geq 5$ or $|I_3| + |I_5| + |I_7| \geq 5$. Assuming the former inequality (the arguments for the latter are similar) we obtain—again from (6.5),

$$\sum_{s \in \{0, 1, 2\}} \left( \sum_{i : \{s, t\} \subseteq S_i} |I_i| \right) \geq 2(|I_2| + |I_4| + |I_6|) \geq 10.$$

Therefore, there must be $s \in \{0, 1, 2\}$ such that

$$\sum_{i : \{s, t\} \subseteq S_i} |I_i| \geq 4,$$

thereby contradicting (6.5).

**Proof of Proposition 7.5:** Assume to the contrary that there is a $(10, 4, 7)$ RS code $C$ over GF($q$), $q$ odd, that contains a failing list $L$ of size 5. By Lemma 7.14, this failing list is a $(4,3)$-configuration with respect to a proper partition vector $P = (I_i)_{i=1}^{10}$, where $|I_i| = 1$ for all $i$. Let $a_1, a_2, \ldots, a_{10}$ be the code locators of $C$. The polynomials $A_{\alpha}(x)$, which are defined by (7.3), can be written, without loss of generality, as

$$A_{\{0, 1\}}(x) = x - a_1, \quad A_{\{0, 1, 3\}}(x) = x - a_2, \quad A_{\{0, 1, 4\}}(x) = x - a_3, \quad A_{\{0, 2\}}(x) = x - a_4, \quad A_{\{0, 2, 3\}}(x) = x - a_5, \quad A_{\{0, 3\}}(x) = x - a_6,$$

$$A_{\{1, 2\}}(x) = x - a_7, \quad A_{\{1, 2, 4\}}(x) = x - a_8, \quad A_{\{1, 3\}}(x) = x - a_9, \quad A_{\{2, 3\}}(x) = x - a_{10}.$$
The polynomials $f_{s,t}(x)$, $0 \leq s < t \leq 4$, are defined accordingly by (7.4).

By Lemma 7.7(b), the ten polynomials $f_{s,t}(x)$ must satisfy (7.5). In particular, for every $0 \leq s < t \leq 4$, the three polynomials $f_{0,s}(x)/g_{0,s,t}(x)$, $f_{0,t}(x)/g_{0,s,t}(x)$, and $f_{s,t}(x)/g_{0,s,t}(x)$, which take the form $(x - \alpha_i)(x - \alpha_j)$, must satisfy the difference condition. By Lemma 7.8, this happens if and only if the code locators satisfy the following six equations:

\[
\begin{align*}
(a_1a_3 - a_7a_9)(a_1 + a_3 - a_4 - a_6) &= (a_3a_9 - a_4a_6)(a_1 + a_3 - a_7 - a_9) \quad (7.21) \\
(a_1a_2 - a_8a_9)(a_1 + a_2 - a_5 - a_6) &= (a_1a_2 - a_5a_6)(a_1 + a_2 - a_8 - a_9) \quad (7.22) \\
(a_2a_3 - a_7a_8)(a_2 + a_3 - a_4 - a_5) &= (a_2a_3 - a_4a_5)(a_2 + a_3 - a_7 - a_8) \quad (7.23) \\
(a_2a_6 - a_7a_{10})(a_2 + a_6 - a_1 - a_5) &= (a_2a_6 - a_1a_5)(a_2 + a_6 - a_7 - a_{10}) \quad (7.24) \\
(a_3a_6 - a_8a_{10})(a_3 + a_6 - a_1 - a_4) &= (a_3a_6 - a_1a_4)(a_3 + a_6 - a_8 - a_{10}) \quad (7.25) \\
(a_2a_4 - a_9a_{10})(a_2 + a_4 - a_3 - a_5) &= (a_2a_4 - a_3a_5)(a_2 + a_4 - a_9 - a_{10}) 
\end{align*}
\]

Defining

\[
\epsilon_7 = (a_3 - a_4)/(a_3 - a_4), \quad \epsilon_8 = (a_2 - a_5)/(a_8 - a_5), \quad \text{and} \quad \epsilon_9 = (a_1 - a_6)/(a_9 - a_6),
\]

equations (7.21)-(7.23) can be re-written as

\[
\begin{pmatrix}
    a_2-a_4 & a_3-a_5 & 0 \\
    a_1-a_4 & 0 & a_3-a_6 \\
    0 & a_1-a_5 & a_2-a_6
\end{pmatrix}
\begin{pmatrix}
    \epsilon_7 \\
    \epsilon_8 \\
    \epsilon_9
\end{pmatrix}
= \begin{pmatrix}
    a_2-a_4+a_3-a_5 \\
    a_1-a_4+a_3-a_6 \\
    a_1-a_5+a_2-a_6
\end{pmatrix}.
\]

(7.27)

Now, if the matrix in (7.27) were nonsingular, then the unique solution of (7.27) would be $\epsilon_7 = \epsilon_8 = \epsilon_9 = 1$, thereby requiring from (7.26) that certain code locators be identical, namely, $\alpha_7 = \alpha_3$, $\alpha_8 = \alpha_2$, and $\alpha_9 = \alpha_1$. Since this is impossible, the matrix in (7.27) must be singular, and this occurs if and only if

\[
-(a_1 - a_5)(a_2 - a_4)(a_3 - a_6) = (a_1 - a_4)(a_2 - a_6)(a_3 - a_5). \quad (7.28)
\]

Re-iterating the analysis, with equations (7.21)-(7.23) now replaced by (7.23)-(7.25), we obtain

\[
-(a_6 - a_5)(a_2 - a_4)(a_3 - a_1) = (a_6 - a_4)(a_2 - a_1)(a_3 - a_5). \quad (7.29)
\]

Subtracting (7.28) from (7.29) and simplifying the result yields

\[
2(a_1 - a_6)(a_2 - a_4)(a_3 - a_5) = 0.
\]

However, this is a contradiction whenever $q$ is odd. We thus conclude that $\mathcal{C}$ cannot contain the failing list $\mathcal{L}$.

\section*{List-10 decoders for $[11, 3, 9]$ RS codes}

\textbf{Proof of Proposition 7.6:} Assume to the contrary that there is an $[11, 3, 9]$ RS code over GF($2^h$) that contains a failing list $\mathcal{L}$ of size 11. By Proposition 6.3 and by property B1 in Proposition 6.5, the failing list corresponds to a symmetric BIBD($11, 5, 2$) (which has 11 blocks), namely it forms a (10, 5)-configuration with respect to a proper partition vector.
\( \mathcal{P} = (I_i) \) such that eleven partition elements \( I_i \) have size 1 whereas all the other partition elements in \( \mathcal{P} \) are empty.

As this BIBD is essentially unique (see \([6, \text{page 73}]\)), we can assume, without loss of
gen\[n]eral\[n] \[n]ity, that the nonempty partition elements in \( \mathcal{P} \) are \( I_i = \{ i \} \), \( 1 \leq i \leq 11 \), where \( S_1, S_2, \ldots, S_{11} \) are given by

\[
S_1 = \{1, 3, 4, 5, 9\}, \quad S_2 = \{2, 4, 5, 6, 10\}, \quad S_3 = \{0, 3, 5, 6, 7\}, \quad S_4 = \{1, 4, 6, 7, 8\}, \\
S_5 = \{2, 5, 7, 8, 9\}, \quad S_6 = \{3, 6, 8, 9, 10\}, \quad S_7 = \{0, 4, 7, 9, 10\}, \quad S_8 = \{0, 1, 5, 8, 10\}, \\
S_9 = \{0, 1, 2, 6, 9\}, \quad S_{10} = \{1, 2, 3, 7, 10\}, \quad S_{11} = \{0, 2, 3, 4, 8\}.
\]

Define \( A_S(x) \) and \( f_{s,t}(x) \) accordingly by (7.3) and (7.4). In particular, we obtain

\[
f_{0,2}(x) = (x - a_9)(x - a_{11}), \quad f_{0,7}(x) = (x - a_3)(x - a_7), \quad f_{0,5}(x) = (x - a_3)(x - a_8), \\
f_{0,10}(x) = (x - a_7)(x - a_8), \quad f_{2,7}(x) = (x - a_5)(x - a_10), \quad f_{2,5}(x) = (x - a_2)(x - a_5), \\
f_{2,10}(x) = (x - a_2)(x - a_{10}).
\]

By Lemma 7.7(b), each of the following sets of three polynomials must satisfy the difference condition: \( \{f_{0,2}(x), f_{0,7}(x), f_{2,7}(x)\} \), \( \{f_{0,2}(x), f_{0,5}(x), f_{2,5}(x)\} \), and \( \{f_{0,2}(x), f_{0,10}(x), f_{2,10}(x)\} \). By Lemma 7.8 we then obtain the following equations on the code locators:

\[
(a_9 + a_{11} - a_3 - a_7)(a_9 a_{11} - a_5 a_{10}) = (a_9 + a_{11} - a_5 - a_{10})(a_9 a_{11} - a_3 a_{7}) \\ (a_9 + a_{11} - a_3 - a_8)(a_9 a_{11} - a_2 a_{5}) = (a_9 + a_{11} - a_2 - a_5)(a_9 a_{11} - a_3 a_{8}) \\ (a_9 + a_{11} - a_7 - a_8)(a_9 a_{11} - a_2 a_{10}) = (a_9 + a_{11} - a_2 - a_{10})(a_9 a_{11} - a_7 a_{8}).
\]

Defining

\[
\epsilon_3 = \frac{(a_{11} - a_3)(a_9 - a_3)}{a_5 - a_3}, \quad \epsilon_7 = \frac{(a_9 - a_7)(a_9 - a_7)}{a_{10} - a_7}, \quad \text{and} \quad \epsilon_8 = \frac{(a_{11} - a_8)(a_9 - a_8)}{a_2 - a_8},
\]

we can re-write (7.30)–(7.32) as

\[
\begin{pmatrix}
1 & 1 & 0 \\
1 & 0 & 1 \\
0 & 1 & 1
\end{pmatrix}
\begin{pmatrix}
\epsilon_3 \\
\epsilon_7 \\
\epsilon_8
\end{pmatrix}
= \begin{pmatrix}
(a_{11} - a_3 + a_9 - a_7) \\
(a_{11} - a_3 + a_9 - a_8) \\
(a_{11} - a_8 + a_9 - a_7)
\end{pmatrix}.
\]

Summing up these equations and recalling that the field size is even, the left-hand side is
identically zero while the right-hand side equals the nonzero value \( a_9 + a_{11} \); hence a
contradiction.

\[\Box\]

### 7.3.5 The low-rate range: Proposition 7.2

**Proof of Proposition 7.2:** The proof is based on Lemma 7.12. One can verify that a
sufficient condition for (7.15) to hold is that \( \{\psi_{s,t}\}_{s,t} \) take either the form \( \psi_{s,t} = \psi_s \psi_t \) or the
form \( \psi_{s,t} = \psi_s + \psi_t \), for some \( \ell+1 \) values \( \psi_0, \psi_1, \ldots, \psi_{\ell} \). The values \( \psi_s \) must form a weak
Sidon set (in the respective group) so as to have distinct values of \( \psi_{s,t} \). We now consider
the two types of polynomials presented in Section 7.3.2, taking \( e \) to be \( k-1 \).

Using polynomials of Type 1 as the \( \binom{\ell+1}{2} \) polynomials \( \{f_{s,t}^{e}(x)\}_{0 \leq s < t \leq \ell} \) in Lemma 7.12,
we require that \((k-1)(q-1)\) and select the respective constant terms \( \psi_{0,1}, \psi_{0,2}, \ldots, \psi_{\ell-1,\ell} \) so
that they satisfy $\psi_{k,t} = \psi_s \psi_t$. The set $\{\psi_0, \psi_1, \ldots, \psi_{\ell}\}$ should be a weak Sidon set of size $\ell+1$ in the multiplicative group of $\text{GF}(q)$. If $\alpha$ is a primitive element in $\text{GF}(q)$ and $\psi_s = \alpha^{\xi_s}$, then an equivalent requirement is that $\{\xi_0, \xi_1, \ldots, \xi_{\ell}\}$ be a weak Sidon set contained in the additive group of $\mathbb{Z}_{(q-1)/(k-1)}$. It is shown in [19, Theorem 1] that this group contains a Sidon set of size $\ell+1$ whenever (7.1) holds (see also [9]).

When using polynomials of Type 2 over $\text{GF}(p^h)$ as $\{f_s(x)\}_{0 \leq s < t \leq \ell}$, we require that $k-1 = p^b$, where $b < h$, and we select the constant terms so that they satisfy $\psi_{k,t} = \psi_s + \psi_t$. The set $\{\psi_0, \psi_1, \ldots, \psi_{\ell}\}$ should be a weak Sidon set of size $\ell+1$ in the range, $R_n$, of a linearized polynomial $\eta(x)$ of degree $p^b$ over $F$ with $p^b$ simple roots in $\text{GF}(p^h)$. This range is an $(h-b)$-dimensional linear space over $\text{GF}(p)$ and is therefore isomorphic to $\mathbb{Z}_{p}^{h-b}$. In [3, Section 5], it is shown that $\mathbb{Z}_{p}^{h-b}$ contains a weak Sidon set of size $\ell+1$ whenever $\ell^{2+o(1)} < p^{h-b}$ (see (7.2)).
Chapter 8

Average number of codewords in list decoding of MDS codes

8.1 Introduction

In Chapter 7, bounds are derived on the largest decoding radius of a given RS code $C$ assuming that the list size does not exceed some predetermined integer value $\ell$. An alternative way to interpret those results is as providing bounds on the maximal list size which is actually realized by the best list decoder with a given radius $\tau$ for $C$. For example, Proposition 7.1 implies that a list decoder of radius 3 for an $[n, n-4, 5]$ RS code over $F = GF(q) , q \geq n$, decodes at least one word $v \in F^n$ into a list of more than $2n/5 - 1$ codewords.

In this chapter we consider another aspect of list decoding performance which is the average number of incorrect codewords that are returned by a list decoder with a given radius $\tau$. By an incorrect codeword we mean a codeword which is different from the transmitted one. The average is taken over all transmitted codewords and error vectors, where the probabilistic model assumed is that of the $q$-ary symmetric channel. An upper bound on that average is derived and specific conclusions are obtained with regard to the performance of Guruswami-Sudan’s list decoders.

We observe here that the analysis made by McEliece and Swanson in their paper [30] with respect to classical RS decoders can be generalized to apply to list decoders which look beyond the classical decoding radius. The results of [30] are described in Section 8.2. The generalization is presented in Section 8.3. We mention that a somewhat similar problem has been considered by Nielsen and Høholdt in [31]. However, the bounds presented here are more explicit and detailed.

8.2 Previous work

In his paper [17], Forney considers decoding of a block code $C$ through one of three decoding strategies which he refers to as (1) hard-decision strategy, (2) error correction and detection, and (3) list decoding.

In each strategy, a decision region $R_m$ is defined for every codeword $c_m$, and the decoder returns $c_m$ iff the received word $v$ falls within the decision region $R_m$. In the first decoding
strategy, the decision regions of distinct codewords are disjoint and they exhaust the space. For every received word \( \mathbf{v} \), the decoder thus returns a unique codeword. A decoding error occurs whenever the codeword returned by the decoder is not the transmitted codeword.

In the second strategy, the decision regions of distinct codewords are disjoint but they do not necessarily exhaust the space. The decoder announces an error detection (without correction) whenever the received word \( \mathbf{v} \) is not included in any of the decision regions. A decoding error occurs if \( \mathbf{c}_m \) is transmitted but \( \mathbf{c}_{m'} \) is the output of the decoder, where \( m' \neq m \).

In the third (list decoding) strategy, the decision regions are not necessarily disjoint and do not necessarily exhaust the space. The decoder’s output is the list of all codewords for which \( \mathbf{v} \in R_m \). A list decoding error occurs if \( \mathbf{c}_m \) is transmitted but does not appear in the decoder’s list. A codeword in the decoder’s list is considered an incorrect codeword if it is not the transmitted codeword.

Forney considers the following two expressions, where \( \text{Prob}(\mathbf{v}|\mathbf{c}_m) \) is the probability of receiving \( \mathbf{v} \) as the output of the channel, given that \( \mathbf{c}_m \) is sent.

\[
P_1 = \frac{1}{|C|} \cdot \sum_m \sum_{\mathbf{v} \in R_m} \text{Prob}(\mathbf{v}|\mathbf{c}_m) \\
P_2 = \frac{1}{|C|} \cdot \sum_m \sum_{\mathbf{v} \in R_m} \sum_{m' \neq m} \text{Prob}(\mathbf{v}|\mathbf{c}_{m'}).
\]

Assume that each of the codewords in \( C \) is transmitted with the same probability. Then \( P_2 \) is the decoding error probability in both the first and the second decoding strategies.

In the second strategy, \( P_1 - P_2 \) is the error detection (without correction) probability. In the list decoding case, \( P_1 \) is the probability of list error, and \( P_2 \) is no longer a probability, but it represents the average number of incorrect codewords in the decoding list. In this case, we may denote this average by \( \bar{N} \), and not by \( P_2 \).

In their paper [30], McEliece and Swanson consider the case where an \([n,k,d]\) MDS code over a finite field \( F = GF(q) \) is used over a channel where error vectors of the same Hamming weight have the same probability to occur. The \( q \)-ary symmetric channel satisfies this property. They assume the model of ‘classical decoding’, namely the second (‘correction-detection’) strategy mentioned by Forney, where the decision region of every codeword \( \mathbf{c} \) is a Hamming sphere of radius \( \tau \leq \lceil \frac{d-1}{2} \rceil \) around \( \mathbf{c} \).

For the above classical decoding model, an upper bound on the error probability \( P_2 \) is computed in [30], as summarized in the following proposition.

**Proposition 8.1** Let \( C \) be an \([n,k,d]\) MDS code decoded by a ‘classical decoder’ of radius \( \tau \leq \lceil \frac{d-1}{2} \rceil \). For every integer \( u \) such that \( 0 \leq u \leq n \), define \( P(u) \) as the probability of a decoding error, given that the error vector is of Hamming weight \( u \). Then

\[
P(u) = 0 \quad \text{for} \quad u \leq d - \tau - 1 \\
P(u) \leq (q-1)^{-(d-1)} \sum_{s=d-u}^{\tau} \binom{n}{s} (q-1)^s \quad \text{for} \quad d - \tau \leq u \leq d - 1 \quad (8.1) \\
P(u) \leq (q-1)^{-(d-1)} V_q(n,\tau) \quad \text{for} \quad u \geq d
\]

where \( V_q(n,\tau) = \sum_{i=0}^{\tau} \binom{n}{i} (q-1)^i \) is the volume of a Hamming sphere of radius \( \tau \).
Weakening the bound in Proposition 8.1 implies that \( P(u) \leq (q - 1)^{-\frac{(d-1)}{\tau!}} \), for every \( 0 \leq u \leq n \). Using the inequality

\[
\mathcal{V}_q(n, \tau) \leq \frac{n^\tau}{\tau!}(q - 1)^\tau, \quad q \geq 4, \quad \tau \geq 2, \quad n \leq q - 1,
\]

and since \( d - 1 > 2\tau \), they conclude that

\[
P_2 \leq \frac{(q - 1)^{-(d-1)}(d-2\tau)}{\tau!} \leq \frac{1}{\tau!},
\]

for every \( 0 \leq u \leq d \). To summarize, for an \([n, k, d]\) MDS code over \( GF(q) \), where \( n < q \), if more than \( \tau = \lfloor (d-1)/2 \rfloor \) errors occur, then the decoding error probability is less than \( 1/\tau! \).

### 8.3 Average number of incorrect codewords in a list

In this section, the results in [30] are generalized to the case where a list decoder is used. An upper bound is derived on the average number \( \mathcal{N} \) of incorrect codewords contained in the decoder’s list. Our results are summarized in Proposition 8.2 below, the proof of which follows the proof of Proposition 8.1 in [30] and adapts it to the case where the decoding radius \( \tau \) may be greater than \( \lfloor \frac{d-1}{2} \rfloor \).

**Proposition 8.2** Suppose an \([n, k, d]\) MDS code \( C \) over \( F = GF(q) \) is used in a channel where error vectors of the same Hamming weight have the same probability to occur and assume that a list decoder is used where the decision region of a codeword \( c \) is the Hamming sphere of radius \( \tau \) around \( c \), where \( \tau \) is any pre-specified positive integer smaller than \( d \). The average number of incorrect codewords in the decoder’s list can be bounded from above by

\[
\mathcal{N} \leq (q - 1)^{-(d-1)}\mathcal{V}_q(n, \tau),
\]

where the average is taken over all transmitted codewords and all possible error vectors.

Lemma 8.3 below summarizes two properties of MDS codes that are used in the proof of Proposition 8.1, as well as in the proof of Proposition 8.2. These two properties are proved in [30].

**Lemma 8.3**

**M1** For an \((n, q^k, d)\) (not necessarily linear) MDS code over \( GF(q) \), the number \( A_u \) of codewords of Hamming weight \( u \), where \( u \geq d \), satisfies \( A_u \leq (n\choose u)(q - 1)^{u-(d-1)} \).

**M2** By puncturing an \([n, k]\) MDS code, leaving only \( n' \) coordinates in every codeword, where \( n' \geq k \), an \([n', k]\) MDS code is obtained.

**Proof of Proposition 8.2:** Due to the linearity of the code \( C \), we assume, without loss of generality, that the zero codeword \( 0 \) is always the transmitted codeword. Given a vector \( v \in F^n \), let us denote by \( \mathcal{N}(v) \) the number of different compositions of \( v \) as \( v = c + e \),
where \( c \) is a nonzero codeword and \( e \in F^n \) is a vector of weight \( \leq \tau \). In other words, \( \mathcal{N}(v) \) is the number of incorrect codewords in the decoder's list, given that the zero codeword 0 is sent and the word \( v \) is received. The notation \( w_H(v) \) will be used to indicate the Hamming weight of a word \( v \). For an integer \( u \) such that \( 0 \leq u \leq n \), let us define

\[
\mathcal{N}_u = \sum_{v : w_H(v) = u} \mathcal{N}(v).
\]

The average number \( \bar{\mathcal{N}}_u \) of incorrect codewords, given that the error vector is of Hamming weight \( u \), can be written as

\[
\bar{\mathcal{N}}_u = \frac{\mathcal{N}_u}{\binom{n}{u}(q-1)^u},
\]

and we therefore aim at bounding \( \mathcal{N}_u \), for every \( 0 \leq u \leq n \).

**Case 1:** \( u \geq d \). For a given error vector \( e \) of weight \( \leq \tau \), the set \( \{ c + e : c \in C \} \) is a coset of an MDS code and is therefore an MDS code itself (though not a linear one). By property M1 in Lemma 8.3, the number of vectors of weight \( u \) in the above set is thus not greater than \( \binom{n}{u}(q-1)^{u-(d-1)} \). For each such vector \( c + e \), the codeword \( c \) is a nonzero codeword because \( e \) is of weight \( \leq \tau \) where \( c + e \) is of weight \( u \geq d > \tau \). We thus obtain:

\[
\mathcal{N}_u \leq \binom{n}{u}(q-1)^{u-(d-1)}V_q(n, \tau), \quad u \geq d.
\]

**Case 2:** \( u \leq d-1 \).

Define \( n' = n - u \). For every set \( I \) of \( n' \) out of \( n \) coordinates (indexes), let us derive an upper bound on the number of different compositions \( c + e \) such that \( c \neq 0 \), \( e \) is of weight \( \leq \tau \), and \( I \) is the set of zero coordinates of the vector \( c + e \). The number \( w \) of nonzero coordinates of \( e \) in \( I \) satisfies \( d - u \leq w \leq \tau \). The right inequality holds because \( e \) is of weight \( \leq \tau \). The left inequality follows from the fact that the code \( C' \) obtained by restricting \( C \) to the coordinates \( I \) is an \([n', k]\) MDS code of minimum Hamming distance \( d - u \). By the property M1 in Lemma 8.3, the number of codewords in \( C' \) of weight \( w \) is at most \( \binom{n'}{w}(q-1)^{w-(d-1)} \). For every codeword \( c \) with \( w \) nonzero coordinates in \( I \), the number of vectors \( e \) of weight \( \leq \tau \) such that \( c + e \) is zero in \( I \) is \( \sum_{s=w}^{\tau} \binom{n'}{s}(q-1)^{s-w} \).

We conclude that in this case,

\[
\mathcal{N}_u \leq \binom{n}{u} \sum_{w=d-u}^{\tau} \binom{n'}{w}(q-1)^{w-(d-1)} \sum_{s=w}^{\tau} \binom{n}{s}(q-1)^{s-w} \leq \binom{n}{u}(q-1)^{u-(d-1)}V_q(n, \tau). \]
In both cases, we get that the average number \( \hat{N}_u \) of incorrect codewords, given that \( u \) errors have fallen, satisfies
\[
\hat{N}_u \leq (q - 1)^{-(d-1)}V_q(n, \tau),
\]
where the upper bound does not depend on \( u \). We can conclude that the average number \( \hat{N} \) of incorrect codewords over all transmitted codewords and all possible error vectors is bounded from above by
\[
\hat{N} \leq (q - 1)^{-(d-1)}V_q(n, \tau).
\]

Now, using the inequality \( V_q(n, \tau) \leq 2^n (q - 1)\tau \), we obtain
\[
\hat{N} \leq 2^n q^{\tau + 1 - d}.
\]
The average \( \hat{N} \) can be bounded from above by \( q^{-\epsilon n} \) if
\[
\tau \leq d - 1 - (1/\log q + \epsilon)n. \tag{8.3}
\]

We thus get that for every \( [n, k, d] \) MDS code, the average number of incorrect codewords while using a list decoder can be made arbitrary small, and in particular smaller than 1, if the decoding radius \( \tau \) is bounded away from the code minimum distance according to (8.3).

We now turn to evaluating the average number of incorrect codewords returned by Guruswami-Sudan’s list decoders. The maximal decoding radius \( \tau \) of their algorithm for an \( [n, k, d] \) RS code is approximately \( n - \sqrt{kn} \), as mentioned in Section 1.3.2. The average number of incorrect codewords in this case does not exceed \( q^{-\epsilon n} \) whenever
\[
n - \sqrt{kn} \leq d - 1 - (1/\log q + \epsilon)n,
\]
or, alternatively,
\[
\sqrt{k/n - k/n - 1/\log q} \geq \epsilon.
\]

For example, when Guruswami-Sudan’s list decoding algorithm is applied to an RS code of length \( n = 256 \) over \( \text{GF}(256) \) at rate \( k/n \approx 0.7 \), the average number of incorrect codewords is not greater than
\[
256^{-(\sqrt{0.7 - 0.7 - 0.125})256} = 6.47 \times 10^{-8}.
\]
The transmitted codeword itself is guaranteed to appear in the decoder’s list if the number of errors does not exceed 41.
Chapter 9

Conclusion

9.1 Contribution of this work

The contribution of this work is concentrated in two main aspects:

I providing efficient procedures for decoding RS codes beyond half the minimum distance, and

II investigating the relations between the decoding radius and the maximal list size of list decoders for RS codes, as well as for general block codes.

9.1.1 Efficient procedures for RS list decoding

We presented already in 1998 (see [36]) a complete (two-stages) quadratic time implementation for Sudan's algorithm. Sudan's algorithm, that was first presented in 1996, is the first polynomial-time algorithm which decodes RS codes beyond half the minimum distance of the code. Our implementation resembles in several aspects the classical decoding algorithm for RS codes given by Berlekamp and Massey:

- Syndrome elements are computed out of the received word. The syndrome formula is a generalization of that used in classical decoding. A bivariate polynomial representation of the syndrome is defined.

- The syndrome elements satisfy an extended key equation in which the unknowns are the coefficients of polynomials with certain degree constraints.

- A set of homogeneous linear equations with a ‘multi-Hankel’ structure is derived from the EKE.

- The number of unknowns in the above equation set equals the number \( \tau \) of correctable errors.

- To solve the equations (and the EKE), we use a generalization of Massey’s algorithm which finds the shortest recurrence relation that is satisfied by the bivariate syndrome polynomial. The time complexity of the solution is quadratic in \( \tau \). The space complexity is linear in \( \tau \).
• The polynomials which solve the EKE are used to compute the consistent codewords.

The first step in our implementation is a quadratic time procedure for interpolating a bivariate polynomial \( Q(x,y) \) from given points \((\alpha_j,v_j)\). The interpolation is feasible if the number of nonzero coefficients assumed in \( Q(x,y) \) is smaller than the number of interpolation points. The bivariate polynomials returned are minimal by means of leading coefficient, where the minimality is determined according to a total order defined between ordered pairs representing degrees in \( x \) and \( y \). The number of unknowns is thus reduced from \( n \) to \( \tau \) in the most time-consuming stage of our interpolation procedure.

The second step in our implementation is a quadratic time procedure for finding all the linear factors \( y - g(x) \) of a given bivariate polynomial \( Q(x,y) \). The procedure is rather simple and has been generalized in [47] to find the roots of univariate polynomials over function fields. An outline for an acceleration of our method has been given in [14] and is claimed to run in time complexity \( O(n \log^2 n) \) (where \( n \) is the code length).

The efficient implementation of Sudan’s algorithm is shown (in Chapter 5) to be extendible into an efficient implementation of Guruswami-Sudan’s algorithm. While the second (factorization) step remains the same, the first step is no longer a simple interpolation of bivariate polynomials. A certain multiplicity \( r \) is forced on the roots \((\alpha_j,v_j)\) of the required polynomial \( Q(x,y) \). Again, we compute syndrome elements and we solve homogeneous linear equations which are different from the original equations formulated in Guruswami-Sudan’s algorithm. The equation system we solve becomes more complicated when \( r > 1 \), and we therefore were unable to reduce the number of unknowns, which thus remains \( r^2 n \). A quadratic time procedure which finds the shortest recurrence relation that is simultaneously satisfied by several bivariate syndrome polynomials is applied to solve our equations.

Decoding of RS codes is typically carried out while reading an encoded data from disks or tapes, and its efficiency is thus very significant. As mentioned, RS decoding algorithms typically apply to other useful codes, such as BCH and conventional (cyclic) RS, as well. Research efforts have therefore been made in the past to develop efficient (quadratic time or less) algorithms for decoding RS codes up to the minimum distance. Enlarging the decoding radius enables to obtain a reduced decoding error probability or, alternatively, to work at higher encoding rates. However, in order to take advantage of these improvements in practical systems, the efficiency of the improved decoding algorithms must remain attractive and must not fall far below that of classical decoding schemes.

To emphasize the applicability of the procedures suggested here, we calculated every operation taken throughout the decoding process and tried to give rather precise expressions for the various time complexities, by including the varying parameters \( \ell \) and \( r \) in the formulas (and not assuming them to be constants).

### 9.1.2 The decoding radius as a function of the maximal list size

The GS lower bound \( \Delta^\text{RS}_l(n,d) \geq [\tau_l(n,d)] - 1 \) is implied by Guruswami-Sudan’s algorithm and is formulated in Section 1.5.1. Justesen and Høholdt indicated in [23] that the GS decoding radius is indeed optimal for some RS codes, and they gave rather specific examples to prove their observation. Our investigation of this bound leads to the following additional conclusions:
The GS bound \( [\tau_\ell(n, d)] - 1 \) is a lower bound on the largest list-\( \ell \) decoding radius of every \((n, M, d)\) block code, and not only of RS codes. Similarly to the well-known lower bound \( \Delta_1(n, d) \geq [(d-1)/2] \), the bound is independent of the alphabet size and applies to non-linear codes as well. It is thus proved by combinatorial arguments only.

The GS bound is shown to be tight for vast ranges of parameters corresponding to RS codes.

However, the GS bound is shown to be improved on in many cases. Some of the improvements apply to every \((n, M, d)\) code and are thus proved combinatorially, while two specific cases applying to RS codes only are synthesized combinatorially and algebraically.

In addition, our analysis of failing list structure implies a characterization of Hamming spheres around vectors in some certain cases and relates it to known combinatorial objects, such as constant-weight codes and BIBDs. Assume the usage of a linear \([n, k, d]\) code over \( F \). Whenever \( \tau_\ell(n, d) \) is an integer, the Hamming sphere of radius \( \tau_\ell(n, d) \) around a vector \( v \in F^n \) of Hamming weight \( < \tau_\ell(n, d) \) does not include more than \( \ell-1 \) nonzero codewords. Otherwise, there would be a failing list of \( \ell+1 \) codewords in which at least one codeword (the zero codeword) is at Hamming distance less than \( \tau_\ell(n, d) \) from \( v \), thus contradicting property N1 in Proposition 6.3. If the Hamming sphere of radius \( \tau_\ell(n, d) \) around a vector \( v' \in F^n \) of Hamming weight \( \tau_\ell(n, d) \) contains \( \ell \) or more nonzero codewords, then all these codewords must have the minimal Hamming weight \( d \). Note that when \( \ell=1 \) and \( \tau_1(n, d) = d/2 \) is an integer, the above observations follow immediately from the triangle inequality satisfied by the Hamming metric.

### 9.2 Problems for further research

As described in Section 1.4.2, several other efficient procedures have been suggested for the implementation of the two stages in Sudan’s and Guruswami-Sudan’s algorithms. It is still unclear which of the procedures will be found as most suitable for real systems, once list decoding becomes a part of RS encoding-decoding standards. Since our interpolation procedure requires the computation of syndrome elements, whereas there are other procedures that do not, it may be necessary to reduce the number of unknowns during the syndrome computation in order to justify it.

A reduction has been obtained in cases where \( r = 1 \), but it is not clear whether it can be achieved in other cases (higher rates) as well. It would be beneficial if the number of unknowns would be made linear in \( \tau \) rather than in \( n \) in high rate cases where the value of \( \tau_\ell(n, d) \) is significantly lower than \( n \). In addition, it should be checked which of the fast interpolation procedures suggested is easily extendible to cases where different multiplicities are forced on different roots, so as to efficiently implement soft-decision decoding of RS codes with large decoding radiiuses, as suggested in [25].

As for the factorization step, though the procedure suggested here is quite simple and efficient, it is still interesting to find out whether the polynomials \( Q^{(B)}(x) \) can be used in some other way so that the error locations and values would be computed directly out of these polynomials, as done in the classical list-1 decoding. Such a method might enable to
output the consistent codewords simultaneously to their reconstruction, as done while using the Chien search and Forney’s algorithm.

In this work we showed that a list-$\ell$ decoding radius greater than the one guaranteed by Guruswami-Sudan’s algorithm can be realized by some RS codes. However, we could only point at an improvement of a single correctable error (enlarging the radius by 1), and we still do not know what is the maximal improvement possible and how close to the GS bound is the upper bound on the decoding radius for the best RS code.

As shown in Chapter 7, RS codes attaining the GS bound were mostly identified in the extreme rate intervals, but were difficult to find in the mid-rate range. The reason may be that the computations are more complicated in the mid-rate range, where the polynomials in the set

$$\bigcup_{s,t} \{ f_{0,s,t}(x)/g_{0,s,t}(x), f_{0,t}(x)/g_{0,s,t}(x), f_{s,t}(x)/g_{0,s,t}(x) \}$$

are not mutually prime. However, it may also be the case that most RS codes in that interval do not attain the GS bound. Since the non-existence of failing lists is also difficult to prove (see the proofs of Propositions 7.5 and 7.6), this question remains open at this point.

The GS lower bound has been shown to hold for every block code. The bound $\Delta_\ell(n,d;q) \geq \lfloor \tau_\ell(n,d) \rfloor - 1$ may be seen as a generalization of the well-known bound $\Delta_1(n,d;q) \geq \lfloor (d-1)/2 \rfloor = \lfloor \tau_1(n,d) \rfloor - 1$. Both bounds do not depend on the alphabet size or on algebraic properties of the code. However, $\lfloor (d-1)/2 \rfloor$ is also an upper bound on $\Delta_1(n,d;q)$ and on the list-1 decoding radius of every code with minimum Hamming distance $d$, whereas $\Delta_\ell(n,d;q)$ has been shown to exceed $\lfloor \tau_\ell(n,d) \rfloor - 1$ in many cases. In addition, if $\Delta_\ell(n,d;q) = \lfloor \tau_\ell(n,d) \rfloor - 1$, it means that there is some $(n,M,d)$ code over GF($q$) with a list-$\ell$ decoding radius $\lfloor \tau_\ell(n,d) \rfloor - 1$, but other codes with the same parameters may have better decoding capabilities. The computation of bounds on the list-$\ell$ decoding radius of the worst, as well as the best, $(n,M,d)$ codes over GF($q$) thus requires further research.

A much more substantial question is whether and how much the RS list decoding radius can be extended beyond $n-\sqrt{kn}$ while keeping the list size polynomial in the code length. Since [23] shows that exponential list sizes are obtained when the radius is very close to $d$, the behavior of the list size when the radius grows from $n-\sqrt{kn}$ to $n-k$ is left to be investigated.

However, we conclude that list decoding is a good suboptimal decoding strategy for RS codes, since pure MLD leads in certain cases to exponential size lists in which all the codewords are at the same distance from the received word (see [23] and Proposition 6.3 in our work). Moreover, by the computations made in Chapter 8, we learn that even when the decoding radius is extended beyond half the minimum distance, a single codeword is obtained in most cases.
Bibliography


