Improved Complexity in Decoding Reed-Solomon Codes

By

Amadou Keita (keita_amadou@student.ceu.edu)

7th June 2019

AN ESSAY PRESENTED TO CENTRAL EUROPEAN UNIVERSITY (CEU) IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF A MASTER OF SCIENCE IN MATHEMATICS AND ITS APPLICATIONS

DECLARATION

This work was carried out at Central European University in partial fulfilment of the requirements for a Master of Science in Mathematics and its Applications.

I hereby declare that except where due acknowledgement is made, this work has never been presented wholly or in part for the award of a degree at Central European University or any other university.

Student: Amadou Keita

Supervisor: Prof. Pal Hegedus

ACKNOWLEDGEMENTS

"And Allah has brought you from the wombs of your mothers when you know nothing. And He gave you hearing and sight and hearts that you might give thanks"(Qur'an 16 : 78)[1]. Therefore, I thank Allah for the grace and guidance, the divine benevolence and the sustenance He accorded me for life. Prof. Pal Hegedus has been the ideal supervisor. His appreciated level of wisdom, quest for quality, insightful criticism, patience and encouragement aided the writing of this material in innumerable ways. I am grateful for his steadfast support both morally and academically. Furthermore, I am grateful for the CEU Master's Excellence Scholarship, and thankful to Prof. Karoly Boroczky, Prof. Laszlo Csirmaz, Elvira Kadvany and Melinda Balazs all of the Mathematics department for their support and insightful conversations, guidance and for their timely response to my needs. I am grateful to Prof. Kathleen E. Lewis, my undergraduate professor, for her invaluable impact on my life. Like my educators, I am most grateful to both my mother and my father for the support, patience and encouragement they accorded me from the first day of my life to my days of academic achievements.

I acknowledge the travel grant from AIMS under the 2017 Post-AIMS Support programme of the African Institute for Mathematical Sciences which enabled me to undertake this research.

DEDICATION

To my loving, patient and determined mother...

Abstract

The Guruswami-Sudan algorithm is a standard algorithm that decodes beyond the classical decoding bound for Reed-Solomon codes. We study some complexity improvement techniques namely polynomial reconstruction and basis transformation which enhance the decoding capabilities of the algorithm, and compare the techniques.

Contents

1. Introduction

Reed-Solomon codes were invented in 1960 and within a decade an efficient algorithm was discovered to decode them. They are efficiently encoded and decoded. Moreover, they are widely used because of their favourable properties including high error correction capability, burst-error correction capability, and erasure-recovery capability.

For Reed-Solomon codes with block length n and dimension k, the Johnson Theorem states that for a Hamming ball of radius $n - \sqrt{nk}$ there can be at most $O(n^2)$ codewords. The first efficient polynomial time algorithm that agrees with the theorem was discovered by Sudan [2], which was improved by Guruswami-Sudan [3]. The Guruswami-Sudan paper [3] considered the following problem: given n distinct points $(x_1, x_2, \cdots, x_n)$ in $\mathbb{F}_q$ and another n points $(y_1, y_2, \cdots, y_n)$ in $\mathbb{F}_q$, find polynomials $f(x)$ of degree at most $k-1$ that agree at at least t points. The algorithm basically takes as inputs a received word $\beta = (\beta_1, \cdots, \beta_n)$ (this is basically some n points $(\alpha_1, \beta_1), \cdots, (\alpha_n, \beta_n)$ where $(\alpha_1, \cdots, \alpha_n)$ are distinct field elements), the number of agreements t and the dimension k of the Reed-Solomon code. The algorithm constructs a bivariate polynomial which is then factored with some conditions. These factors, when they satisfy (as equations) at least t of the n points, are listed. A list of those polynomials identify with codewords yielding a full list of possible codewords that might have been transmitted. Enumerating all possible codewords this way suggests the name list decoding. A quest for efficient decoding in a larger radius, or with fewer polynomials listed, or enumerating polynomials in a reduced time is keeping researchers busy.

In [4], a polynomial reconstruction technique was used to improve on the decoding capability of the Guruswami-Sudan algorithm. It was established that the Guruswami-Sudan algorithm can list decode beyond the Johnson radius. In this reconstruction process, we apply some nice reduction technique and then apply the Guruswami-Sudan algorithm appropriately to the problem. This guarantees efficient decoding with a larger radius.

The idea of improving Sudan [2] and Guruswami-Sudan [3] by using some transformation or re-encoding was explored by many researchers. Nevertheless, the basis transformation that we considered actually incorporates a transformation where the n points get modified in a certain way and then re-encoded. In the basis transformation, we write the Guruswami-Sudan algorithm in terms of modules over a univariate polynomial ring, and then try to get some reduction in the total time required to complete the interpolation step of the Guruswami-Sudan algorithm. If the length of the list of polynomials to be enumerated exceeds the multiplicity of the bivariate polynomial, we attain an improvement in the decoding time of the interpolation step of the Guruswami-Sudan algorithm.

In the rest of the work, we give a precise literature review in Chapter 2, discuss preliminary concepts in Chapter 3 and provide a detailed treatment of the Guruswami-Sudan interpolation in Chapter 4. Furthermore, we study polynomial reconstruction in Section 5.2 and basis transformation in Section 5.3 for improved complexities in Chapter 5. We give a short conclusion in Chapter 6.

2. Literature Review

Reed-Solomon codes are useful in many ways because of their efficient error correction property. Classically, algorithms were constructed to decode Reed-Solomon codes but only up to a certain radius. In the year 1997, Sudan [2] built on the work of Welch-Berlekamp [5] and Ar et al. [6] to change the dynamics of decoding Reed-Solomon codes by introducing an algorithm that efficiently decodes beyond the classical decoding bound. In the year 1999, the famous Guruswami-Sudan algorithm [3] which improved on the Sudan algorithm [2] was published. The Guruswami-Sudan algorithm [3] is termed the real game changer in decoding Reed-Solomon codes due to its sophisticated nature and capacity. The algorithm's complexity, just as Sudan's [2], is in polynomial time.

The Guruswami-Sudan algorithm [3] executes the decoding of Reed-Solomon codes in two steps - bivariate polynomial interpolation and factorisation. Some later authors considered ways of improving the complexity of solving the Guruswami-Sudan problem [3]. In [7], a new code was constructed on which the performance of the Guruswami-Sudan algorithm attains better (optimal) complexity instead of attempting to improve the Guruswami-Sudan algorithm. We will not discuss new construction of codes since a case like this is rare. In the more frequent cases, some complexity in solving the Guruswami-Sudan problem is improved in either the interpolation or the factorisation step.

In [4], an algorithm for polynomial reconstruction that considers the Guruswami-Sudan problem [3] in instances and then solve them by applying the Guruswami-Sudan algorithm an appropriate number of times was given. This approach registers an increase in the decoding radius, hence an improved complexity was attained.

Several authors tried to improve the complexities in the interpolation step of the Guruswami-Sudan algorithm. In the work of Kotter [8], Roth-Ruckenstein [9] and others, transformation and re-encoding were discussed. Roth-Ruckenstein [9] reformulated the problem in Sudan [2] into a key equation over a univariate polynomial ring. The Guruswami-Sudan problem was reformulated as a system of key equations in [10], [11] and [12]. Alekhnovich's algorithm [13] is based on a conversion of Grobner bases from one ordering to another using divide-and-conquer and [14] presented a list decoding of Reed-Solomn codes from a Grobner basis perspective. Beelen and Brander [12] used key equations to describe the interpolation step in terms of modules over a univariate polynomial ring. This approach resulted in an improved complexity for the interpolation step of the Guruswami-Sudan problem.

In this thesis, we study the Guruswami-Sudan algorithm for decoding Reed-Solomon codes. We consider polynomial reconstruction [4] and basis transformation [12] for improved complexities.

In the next chapter, we discuss preliminary concepts for understanding the Guruswami-Sudan interpolation.

3. Preliminary Concepts

In Chapter 2, we highlighted some works conducted on decoding Reed-Solomon codes. In this chapter, we discuss concepts required for the understanding and decoding of Reed-Solomon codes beyond the classical decoding bound. We present a fair idea of coding in general, introduce finite fields and outline some uses of Reed-Solomon codes.

3.1 Definitions

Let Σ be an alphabet of n distinct elements. We will let $\mathbb{F}_q = \Sigma$ until otherwise stated, since Reed-Solomon codes are linear codes. We make the following definitions:

Definition 3.1.1 (Hamming distance [15])**.** Let $x = (x_1, x_2, \cdots, x_n)$ and $y = (y_1, y_2, \cdots, y_n)$ be two strings in $\mathbb{F}_q^n$. The number of inequal symbols

$$d(x, y) = |\{i \leq n | x_i \neq y_i\}|$$

is the Hamming distance between strings x and y.

Definition 3.1.2 (Hamming weight [15])**.** Let $x = (x_1, x_2, \cdots, x_n) \in \mathbb{F}_q^n$ and $\mathbf{0} = (0, 0, \cdots, 0) \in \mathbb{F}_q^n$. The Hamming distance $d(x, \mathbf{0})$ is called the Hamming weight.

Definition 3.1.3 (Code [15])**.** The subset $\mathcal{C}$ of $\mathbb{F}_q^n$ is called a q-ary code. The elements of $\mathcal{C}$ are called codewords. If $q = 2$, then $\mathcal{C}$ is a binary code.

Definition 3.1.4 (q-ary code [16])**.** Let $\mathbb{F}_q$ be a finite set of q elements. A set of strings of length n form a q-ary code $\mathcal{C}$ if their entries are from $\mathbb{F}_q$.

Definition 3.1.5 (Field [17])**.** A field is a commutative ring in which every non-zero element has a multiplicative inverse.

Definition 3.1.6 (Reed-Solomon code [18])**.** Let $(x_1, \cdots, x_n)$ be n distinct elements in $\mathbb{F}_q$. The map

$$\begin{aligned} \Psi : \mathbb{F}_q^k &\longrightarrow \mathbb{F}_q^n \\ (f_0, f_1, \cdots, f_{k-1}) &\longmapsto (f(x_1), f(x_2), \cdots, f(x_n)) \end{aligned}$$

where $(f_0, f_1, \cdots, f_{k-1})$ is a k-dimensional vector such that

$$f(x) = f_0 + f_1 x + \cdots + f_{k-1} x^{k-1} \in \mathbb{F}_q[x]$$

is some encoding process. The image space of this linear map forms a Reed-Solomon code.

3.2 Error Correction

A message is a sequence of codewords. It is transmitted through a noisy channel. When a codeword is transmitted, it can happen that something different is received. We call this the received word. The received word is obviously an element of the vector space $\mathbb{F}_q^n$ but might not be a codeword. That is, during transmission, errors can take place. The error can be an erasure (i.e. to have a message symbol deleted), alteration (i.e. to have a message symbol replaced with another message symbol) or concatenation (i.e. to have codeword length extended). Note that the Guruswami-Sudan algorithm (discussed in Chapter 4) addresses the cases of alterations and erasures. We will consider the case of alterations in this section for linear codes in general. A linear code, $\mathcal{C}$ is a code with the property that any linear combination of codewords is also a codeword. In other words, $\mathcal{C} \leq \mathbb{F}_q^n$ is a subspace. We present two different types of error correcting codes below.

The basic idea of error correction is to see to it that when a codeword is transmitted through a channel, which could influence limited alterations on the transmitted symbols, the receiver should successfully decode the transmitted message. Given a code $\mathcal{C} \subset \mathbb{F}_q^n$, the number of codewords of $\mathcal{C}$ is at most q^n. Our goal here is to show how $\mathcal{C}$ could be built for it to be an error correcting code. Consider the code $\mathcal{C} = \mathbb{F}_2^5$. Unfortunately, this cannot correct even one error. For example, if the codeword 10000 is sent but the channel transmits 11000 the receiver could decode the message to be any of 10000, 01000, 11100, 11000, 11010, 11001 even if we assume that there is at most one error. Therefore, we see that $\mathcal{C}$ is not a good error correcting code.

Now, let $\mathcal{C} \subset \mathbb{F}_2^5$ with codewords

$$\mathcal{C} = \{00000, 00111, 11001, 11110\}.$$

If a codeword is transmitted from $\mathcal{C}$ with at most one error, the receiver will know the actual message sent. In this case, we have just 4 codewords in our code but it can be massive. Observe that each non-zero codeword in $\mathcal{C}$ has Hamming weight at least 3. Because the code is linear, it implies that the minimum (Hamming) distance between any two codewords is at least 3.

Definition 3.2.1. A linear code $\mathcal{C}$ is an $[n, k, d]_q$ code if $\mathcal{C}$ has dimension k, minimum distance d and each codeword of $\mathcal{C}$ has length n.

Theorem 3.2.2. *Let $\mathcal{C}$ be an $[n, k, d]_q$ linear code. The number of errors e that can be corrected by $\mathcal{C}$ is smaller than half the minimum distance of $\mathcal{C}$.*

Proof. By way of contradiction, suppose y is a received vector and that there are codewords $x_1, x_2 \in \mathcal{C}$ such that $d(y, x_1) \leq \dfrac{d-1}{2}$ and $d(y, x_2) \leq \dfrac{d-1}{2}$. By the triangle inequality, this implies that

$$d(x_1, x_2) \leq d(x_1, 0) + d(x_2, 0) \leq \frac{d-1}{2} + \frac{d-1}{2} = d - 1 < d.$$

This is a contradiction since $d(x_1, x_2) = d(x_1 - x_2, 0) \geq d$. □

In information transmission, we prefer codes with large minimum distance simply because we want to get more errors corrected.

Example 3.2.3. For $\mathcal{C} = \{00000, 00111, 11001, 11110\}$, we know $d = 3$. Hence, $\mathcal{C}$ can correct at most one error since $2e \leq d - 1$. If $x \in C$ is transmitted and

1. $w = 11001$ is received, we see that x must be 11001 in which case an error was impossible.
2. $y = 11101$ is received, there must be an error. By inspection, $x = 11001$ is the original.
3. $z = 10100$ is received, then there is no codeword at distance 0 or 1 from it. The codeword x could be either 00000 or 11110.

Definition 3.2.4 (Uniquely decodable [16])**.** A code $\mathcal{C}$ is uniquely decodable if for any finite source sequence, the sequence of code symbols corresponding to this source is different from the sequence of code symbols corresponding to any other source sequence.

Remark 3.2.5. Another technique of building error correcting codes is to construct codewords such that no codeword is a prefix of the other. By Definition 3.2.1, uniquely decodable codes are not (linear) codes. They are the kind of messages use for information transmission.

3.3 Encoding, Error Detection and Decoding

There are many known techniques of encoding information which is then transmitted via a channel in which the encoded message might encounter some alterations. The alterations can be a change in the symbols of the codewords or that some symbols are erased completely. We will discuss a few of these techniques that are connected to Reed-Solomon codes.

3.3.1 Parity check. This is a technique of encoding codewords of a code $\mathcal{C} \subset \mathbb{F}_2^n$ where the encoder adds a parity check digit mod 2 to each codeword. The parity of a codeword is even or odd (this is basically the number of the symbol 1 in a codeword). The technique of encoding codewords of a code $\mathcal{C} \subset \mathbb{F}_2^n$ where the encoder adds a parity check digit to each codeword is referred to as a Simple Parity Check or a One-dimensional Parity Check. The encoder can decide to add a parity check mod 2 to each codeword, which is just ensuring that the number of the symbol 1 in every message string is even. This type of One-dimensional Parity Check can be referred to as even-parity check. It works similarly if the encoder chooses to make sure that the number of the symbol 1 in each message string is odd. In this case, a 0 parity check digit is added to codewords with odd number of the symbol 1 and a 1 parity check digit is added to codewords with even number of the symbol 1. The type of One-dimensional Parity Check where each message string contains odd number of the symbol 1 can be referred to as odd-parity check. However, a linear combination of its encoded strings does not give another encoded string.

In a One-dimensional Parity Check, the encoder adds a parity check digit accordingly and then transmits the encoded message. The receiver computes the parity of the received

message to determine if there occurred an error. In the case of even-parity check, if the receiver determines that the parity of the received message is odd, the receiver will reject the message or ask for a re-transmission on the basis that at least one symbol was altered during transmission. Otherwise, the receiver assumes that there were no alterations. One-dimensional Parity Check only detects an error but cannot correct with certainty.

Like One-dimensional Parity Checks, a Two-dimensional Parity Check adds a parity check digit to every codeword accordingly. Even more, a Two-dimensional Parity Check computes parity check digits for columns that resulted from an alignment of all encoded strings of the One-dimensional Parity Check. For example, the code below is a demonstration of a Two-dimensional Parity Check. The parity-check digits are recorded in the last column for each row, and the bottom row for each column.

$$\begin{matrix}0&1&1\\1&0&1\\1&1&0\end{matrix} \xrightarrow[\text{Parity}]{\text{Encode}} \left[\begin{array}{ccc|c}0&1&1&0\\1&0&1&0\\1&1&0&0\\\hline 0&0&0&0\end{array}\right] \xrightarrow[\text{Channel}]{\text{Transmission}} \left[\begin{array}{ccc|c}0&1&1&0\\1&0&0&0\\1&1&0&0\\\hline 0&0&0&0\end{array}\right] \xrightarrow[\text{Decoding}]{\text{Detection}} \left[\begin{array}{ccc|c}0&1&1&0\\1&0&\boxed{0}&0\\1&1&0&0\\\hline 0&0&0&0\end{array}\right]$$

Suppose an error in transmission is experienced. It is detected by observing the parities across the rows and down the columns. The intersection of a row and a column whose parities do not check out will be the coordinate of the error symbol.

The Two-dimensional Parity Check improves on the One-dimensional Parity Check for better error detection. It detects two errors and corrects only one error.

Definition 3.3.2 (Burst error)**.** The type of error in which two or more symbols in a string of $\mathbb{F}_2^n$ change from 0 to 1 or vise-versa is called burst error. The length of the burst error is the number of string symbols from the first corrupted to the last corrupted symbol.

In simple parity check with strings of length r, a length of r^2 message is transmitted whereas a length of $(r+1)^2$ message is transmitted in the Two-dimensional case. With a Two-dimensional Parity Check, the likelihood of detecting burst errors increases. However, if four symbols of the message are altered a Two-dimensional Parity Check will not correct an error.

3.3.3 Syndrome. The idea of encoding messages to be transmitted over a channel is basically to improve the decoding capabilities since alterations might occur during transmission. There are different ways to do this. Consider constructing a code with codewords having some of their index elements dependent on other index elements. One family of such codes is repetition codes. In a repetition code, every codeword has message digits and check digits. The message digits is in fact the information the encoder desires to transmit, and the repetition of the message digits for a number of times makes its check digits. A codeword, in this case, is the concatenation of the message digits and the check digits. Another interesting family is the parity check codes. We have discussed inclusion of parity check codes in Subsection 3.3.1 and we have already remarked there that that is not a simple parity check problem. A simple parity check can be referred to as a single-parity-check code. In a single-parity-check code, the parity of the message digits are considered and it is concatenated with

the message digits to form codewords.

In order to make the decoding process fast, a better technique is desired. For both repetition and single-parity-check codes, we saw that the check digits are a function of the message digits. Therefore, presenting a general case where the check digits are a function of the message digits is necessary for gains in efficient and fast decoding.

Definition 3.3.4 (Parity-Check Matrix $\mathcal{H}$)**.** This is a matrix whose rows form a basis for the dual space $\mathcal{C}^{\perp} \leq \mathbb{F}_q^n$. It is a $k \times n$ matrix.

Definition 3.3.5 (Syndrome [19])**.** The syndrome $\mathbf{s}$ of any received vector r is defined by the equation $\mathbf{s} = \mathcal{H}r$, where $\mathcal{H}$ is the parity-check matrix. See Example 3.3.6.

Example 3.3.6. Consider a code in which the length of codewords is 10; 5 message digits and 5 check digits. Denote m_1, m_2, m_3, m_4, m_5 as the message digits and $m_6, m_7, m_8, m_9, m_{10}$ as the check digits, and set the parity-check equations as follows:

$$\begin{aligned} m_6 &= m_1 + m_2 + m_3 + m_4 \\ m_7 &= m_1 + m_2 + m_3 + m_5 \\ m_8 &= m_1 + m_2 + m_4 + m_5 \\ m_9 &= m_1 + m_3 + m_4 + m_5 \\ m_{10} &= m_2 + m_3 + m_4 + m_5 \end{aligned}$$

These equations form the system

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ \\ m_{10} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \\ 0 \end{bmatrix}. \tag{3.3.1}$$

Let $\mathcal{H}$ be the coefficient matrix in Equation (3.3.1). We call $\mathcal{H}$ the *parity-check matrix*. The code $\mathcal{C}$ has 2^5 codewords, and a received vector is a codeword only when it satisfies Equation (3.3.1). A decoder identifies and corrects errors by first calculating *syndrome* digits (hence the name *syndrome decoding*) using Equation (3.3.1) as:

$$\begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_5 \end{bmatrix} = \mathcal{H} \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_{10} \end{bmatrix}, \tag{3.3.2}$$

where $\mathbf{s} := [s_1, s_2, \cdots, s_5]^T$ is the *syndrome* vector and $r := [r_1, r_2, \cdots, r_{10}]^T$ is the received vector. The *syndrome* digits inform the decoder of the pattern of parity-check failures on the received vector. Because of linearity, the decoder only needs to check the codewords that have the same *syndrome* as the received vector. This simplifies the task since knowledge

of the error pattern automatically relates to what the transmitted codeword was, and it is given by the relation

$$y = x + r$$

where x was transmitted, r is the error pattern and y is received.

Definition 3.3.7 (Generator matrix [15])**.** Let $\mathcal{C}$ be an $[n, k, d]_q$ Reed-Solomon code. A $k \times n$-matrix whose rows form a basis of $\mathcal{C}$ is called a generator of $\mathcal{C}$. Linear combinations of the rows of a generator matrix form the image space of $\mathcal{C}$. Since $\mathcal{C}$ is a vector space (a subspace of $\mathbb{F}_q^n$), a basis of $\mathcal{C}$ is just a maximal set of linearly independent vectors and the dimension of $\mathcal{C}$ is the cardinality of this basis (the number of rows of a generator matrix).

Theorem 3.3.8 (See Lemma 2.14 in [15])**.** *Denote $\mathcal{G} = (I|P)$, a generator matrix of an $[n, k, d]_2$-code. Then the check matrix is given by $\mathcal{H} = (P^T|I)$, where I in $\mathcal{G}$ is the $k \times k$-unit matrix and I in $\mathcal{H}$ is the $(n-k) \times (n-k)$-unit matrix and P^T is the transpose of P.*

Proof. Since $\mathcal{H}$ is the check matrix, every codeword is orthogonal to every row of $\mathcal{H}$. We know the rows of $\mathcal{G}$ are codewords (in fact, their linear combinations give all codewords). Therefore, we want to show that every row of $\mathcal{G}$ is orthogonal to every row of $\mathcal{H}$:

$$\mathcal{G}\mathcal{H}^T = (I|P)(P^T|I)^T = P + P = 0 \text{ matrix.}$$

□

3.4 Finite Fields

From Definition 3.1.5, it is straightforward to see that a field has at least two elements. We denote $\mathbb{F}_q$ as a field with q elements. The construction and error-correction of Reed-Solomon codes depend on the arithmetic of finite fields. For this reason, it will be ideal to understand some basics of finite fields.

Finite fields are constructed. Consider $\mathbb{F}_3 = \{0, 1, 2\}$. This is a field of 3 elements. See the addition and multiplication of its elements below.

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

*	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

From the tables, we see that for any elements $\alpha, \beta, \gamma \in \mathbb{F}_3$, $\alpha + \beta = \beta + \alpha$, $\alpha * \beta = \beta * \alpha$, $\alpha * (\beta + \gamma) = \alpha * \beta + \alpha * \gamma$, and if $\alpha \neq 0$ and $\beta \neq 0$, then $\alpha * \beta \neq 0$. Now, consider another set $\mathbb{E}_4$ of a larger cardinality. We know any number modulo 4 belongs to the set $\mathbb{E}_4 = \{0, 1, 2, 3\}$. However, $2 \in \mathbb{E}_4$ with $2 * 2 = 0 \bmod 4$. A non-zero element in this set has no multiplicative inverse. So $\mathbb{E}_4$ is simply not a field. Nevertheless, we can construct a field using $\mathbb{E}_4$. Write

$\mathbb{E}_4$ as $\mathbb{F}_{p^r}$ with p a prime (i.e. $\mathbb{F}_{2^2}$ where $r = 2$ and $p = 2$). We need a monic irreducible polynomial of degree $r = 2$ with coefficients in $\mathbb{F}_p = \mathbb{F}_2$. Since $-1 = +1$ and any multiple of 2 equals 0, the only possibilities are

$$x^2 + x + 1, \quad x^2 + x, \quad x^2 \quad \text{and} \quad x^2 + 1.$$

Note that the only irreducible among these polynomials is $x^2 + x + 1$. So we define

$$f(x) := x^2 + x + 1.$$

If there exists ω such that $f(\omega) = 0$, then $\omega^2 + \omega + 1 = 0$. We have $\omega^2 = -\omega - 1 = \omega + 1$. From this, we get $\omega^3 = \omega^2 + \omega = 1$. Then $\mathbb{E}_4 = \{0, 1, \omega, \omega+1\}$ gives us a new set of elements. See the addition and multiplication of these elements below.

$+$	0	1	ω	$\omega+1$
0	0	1	ω	$\omega+1$
1	1	0	$\omega+1$	ω
ω	ω	$\omega+1$	0	1
$\omega+1$	$\omega+1$	ω	1	0

$*$	0	1	ω	$\omega+1$
0	0	0	0	0
1	0	1	ω	$\omega+1$
ω	0	ω	$\omega+1$	1
$\omega+1$	0	$\omega+1$	1	ω

From the current construction, we have a field of 4 elements.

In general, if q is a prime power we construct a field $\mathbb{F}_q$ in the following steps:

- ⋆ write $\mathbb{F}_q = \mathbb{F}_{p^r}$ for some p prime,
- ⋆ find a monic irreducible polynomial of degree r with coefficients in $\mathbb{F}_p$, set

$$f(x) = x^r + a_{r-1}x^{r-1} + \cdots + a_0,$$

- ⋆ let $f(\omega) = 0$ such that $\omega^r = -a_{r-1}\omega^{r-1} - \cdots - a_0$,
- ⋆ compute the elements of $\mathbb{F}_q$ by multiplying ω^r by increasing powers of ω.

In this chapter, we have learned some important concepts including the significance of finite fields in construction Reed-Solomon codes. We have also seen that polynomials identify with codewords, and this will be helpful in reformulating the Guruswami-Sudan problem to simply solve it. In Chapter 4, we will strengthen the foundation specifically for solving the Guruswami-Sudan problem.

4. Guruswami-Sudan Interpolation

In Chapter 3, we presented some important concepts that form prerequisites to the understanding of our main task. In this chapter, we use the definition of a Reed-Solomon code, present a nice way to write any bivariate polynomial as a univariate polynomial, and we discuss Hasse derivatives which is core is solving most of our problems in Chapter 5.

4.1 Reed-Solomon Codes

Let us use Definition 3.1.6 to construct an example of a Reed-Solomon code. Since we have shown how a four-element field is constructed in Section 3.4, we will like to use that field as our alphabet.

Example 4.1.1. Consider $\mathbb{F}_4 = \{0, 1, \omega, \omega^2\}$ constructed using a quadratic polynomial

$$x^2 + x + 1 \in \mathbb{F}_2[x]$$

with coefficients in $\mathbb{F}_2$. We write $x^2 = 1 \times x + 1 \times 1$ resulting in a $(k = 2)$-dimensional vector $(f_0, f_1) = (1, 1)$. The polynomial $f(x) = f_0 + f_1 x$ is the map

$$(f_0, f_1) \longmapsto ((f_0 + f_1 0), (f_0 + f_1 1), (f_0 + f_1 \omega), (f_0 + f_1 \omega^2)),$$

which has all together 4^2 codewords. We enumerate them below.

(f_0, f_1)	Evaluation	(f_0, f_1)	Evaluation
$(0, 0)$	$(0, 0, 0, 0)$	$(\omega, 0)$	$(\omega, \omega, \omega, \omega)$
$(0, 1)$	$(0, 1, \omega, \omega^2)$	$(\omega, 1)$	$(\omega, \omega^2, 0, 1)$
$(0, \omega)$	$(0, \omega, \omega^2, 1)$	(ω, ω)	$(\omega, 0, 1, \omega^2)$
$(0, \omega^2)$	$(0, \omega^2, 1, \omega)$	(ω, ω^2)	$(\omega, 1, \omega^2, 0)$
$(1, 0)$	$(1, 1, 1, 1)$	$(\omega^2, 0)$	$(\omega^2, \omega^2, \omega^2, \omega^2)$
$(1, 1)$	$(1, 0, \omega^2, \omega)$	$(\omega^2, 1)$	$(\omega^2, \omega, 1, 0)$
$(1, \omega)$	$(1, \omega^2, \omega, 0)$	(ω^2, ω)	$(\omega^2, 1, 0, \omega)$
$(1, \omega^2)$	$(1, \omega, 0, \omega^2)$	(ω^2, ω^2)	$(\omega^2, 0, \omega, 1)$

It is simpler and ideal for the sake of space and time to write a code in the form of a generator matrix than to enumerate every codeword like we did in the table above. The generator matrix for this code is

$$\mathcal{G} := \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & \omega & \omega^2 \end{pmatrix}$$

Example 4.1.1 is a $[4, 2, 3]_4$ Reed-Solomon code. Its minimum distance is 3 because every non-zero codeword has a Hamming weight at least 3. In General, Reed-Solomon codes have minimum distance $d = n - k + 1$, and they can correct up to

$$e = \left\lfloor \frac{n - k}{2} \right\rfloor$$

errors in classical decoding (which is at most one error in Example 4.1.1).

Theorem 4.1.2. *If a polynomial of degree at most $k-1$ has at least k roots, then this polynomial is the zero polynomial.*

Proof. Take $R(x) \in \mathbb{F}[x]$ with $\deg R(x) \leq k-1$ and assume that $R(x)$ has at least k roots. Suppose $\exists$ an $\tilde{x}$ such that $R(\tilde{x}) = 0$. Then $(x-\tilde{x}) \mid R(x)$. If $R(x)$ has $(k-1)+1 = k$ distinct roots $\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_k$, then

$$(x-\tilde{x}_1)(x-\tilde{x}_2)\cdots(x-\tilde{x}_k) \mid R(x).$$

We have a degree k polynomial that forms a factor of a degree at most $k-1$ polynomial. Therefore, $R(x)$ must be a zero polynomial. □

Theorem 4.1.2 implies that $d = n-k+1$ is a lower bound for the minimum distance of a Reed-Solomon code. In the next example, we discuss how a Reed-Solomon code could be written as a generator matrix first by writing field elements as powers of one field element, called a primitive element.

Example 4.1.3. Let the field

$$\mathbb{F}_8 = \{0, 1, \omega, \omega^2, \omega^3, \omega^4, \omega^5, \omega^6\}$$

be constructed from the irreducible polynomial $x^3 + x + 1$ with coefficients from $\mathbb{F}_2$. Note that ω here is not the same as in Example 4.1.1, and that this example is suitable for the case when the generator of the field is primitive. So we let ω be a root of this polynomial, then $\omega^3 = \omega + 1$. We construct a generator matrix below with the monomials on the left and their evaluation as a row of the matrix on the right.

$$\begin{array}{ccc} 1 & \longrightarrow & \\ x & \longrightarrow & \\ x^2 & \longrightarrow & \\ x^3 & \longrightarrow & \end{array} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 \\ 0 & 1 & \omega^2 & \omega^4 & \omega^6 & \omega & \omega^3 & \omega^5 \\ 0 & 1 & \omega^3 & \omega^6 & \omega^2 & \omega^5 & \omega & \omega^4 \end{pmatrix}$$

From this construction, we can generate the complete Reed-Solomon code by computing the linear combinations of the rows of the above matrix.

A very interesting fact about Reed-Solomon codes is that their generator matrices are contained in each other. From the generator matrix in Example 4.1.3, we define the following:

$$\mathcal{G}_1 := \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \quad \mathcal{G}_2 := \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 \end{pmatrix}$$

$$\mathcal{G}_3 := \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 \\ 0 & 1 & \omega^2 & \omega^4 & \omega^6 & \omega & \omega^3 & \omega^5 \end{pmatrix} \quad \mathcal{G}_4 := \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 \\ 0 & 1 & \omega^2 & \omega^4 & \omega^6 & \omega & \omega^3 & \omega^5 \\ 0 & 1 & \omega^3 & \omega^6 & \omega^2 & \omega^5 & \omega & \omega^4 \end{pmatrix}.$$

The matrices $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4$ are the generator matrices for $[8,1,8]_8$, $[8,2,7]_8$, $[8,3,6]_8$ and $[8,4,5]_8$ Reed-Solomon codes, respectively. Note that if there are more than one irreducible polynomials, the choice does not influence the resulting code. This is simply because the polynomial only helps in constructing the field elements, which are fixed, and determining the dimension of the generator matrix which is the same for each possible choice of irreducible polynomial since they have the same degree. Therefore, choosing $x^3 + x^2 + 1$ instead of $x^3 + x + 1$ will make no difference.

Now, we demonstrate the identification of polynomials with codewords.

Example 4.1.4. Given the generator matrix of a $[7,4,4]_7$ Reed-Solomon code as

$$\mathcal{G} := \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 1 & 4 & 2 & 2 & 4 & 1 \\ 0 & 1 & 1 & 6 & 1 & 6 & 6 \end{pmatrix},$$

we want a polynomial of degree at most $k-1$ (i.e. 3). Note that such a polynomial can have at most 3 roots. Let the polynomial be

$$(x-1)(x-2)(x-3) = x^3 + x^2 + 4x + 1.$$

As explained in Example 4.1.3, this is just row 4 of $\mathcal{G}$ plus row 3 of $\mathcal{G}$ plus 4 times row 2 of $\mathcal{G}$ plus row 1 of $\mathcal{G}$. Therefore, the polynomial is identified with the codeword $(1,0,0,0,6,3,4)$.

From the examples above, we learned how finite fields are constructed with the knowledge of irreducible polynomials, and how polynomials identify with codewords of a Reed-Solomon code. Suppose $c_1 \in C$ such that $c_1 = (f(x_1), \cdots, f(x_n))$ is transmitted with $f(x)$ being the original polynomial of degree at most $k-1$ (where k is the dimension of $\mathcal{C}$) but $c_2 = (y_1, \cdots, y_n)$ is the received vector. Let $p(x)$ be a polynomial with degree at most $k-1$ such that

$$| \{i \leq n \, | p(x_i) \neq y_i\} | \leq e.$$

The Guruswami-Sudan algorithm finds the polynomial $p(x)$ (which eventually turns out to be $f(x)$) in two steps:

⋆ INTERPOLATION: Let $c_2 = (\beta_1, \cdots, \beta_n)$ be the received message. Construct a bivariate polynomial

$$Q(x, y) = \sum_{i,j \geq 0} q_{i,j} x^i y^j \tag{4.1.1}$$

such that

- $Q(x, y) \neq 0$;
- $Q(x, y)$ has a zero of multiplicity at least m at each of (α_i, β_i) where $x_i = \alpha_i$ $\forall i$;
- and the $(1, k-1)$-weighted degree of Q is as small as possible.

⋆ FACTORIZATION: Find all factors of $Q(x, y)$ of the form $y - p(x)$ such that the degree of $p(x)$ is at most $k - 1$. Collect these as

$$\mathcal{L} = \{p_1(x), p_2(x), \ldots, p_l(x)\}. \tag{4.1.2}$$

See the original algorithm and its interpolation and factorization theorems in [3] and some explanations of them in [18]. In Example 4.1.6, we show how the Guruswami-Sudan algorithm works.

Proposition 4.1.5 ([3])**.** Given n pairs of points as in the Guruswami-Sudan problem, $\{(\alpha_i, \beta_i)\}_{i=1}^n \in \mathbb{F}_q \times \mathbb{F}_q$, there exists a bivariate polynomial $Q(x, y)$ as sought in the interpolation step such that $Q(\alpha_i, \beta_i) = 0$ for all $i = 1, \cdots, n$ provided

$$n\binom{m+1}{2} < \frac{l(l+2)}{2(k-1)}$$

(and it can be found by solving a linear system).

Proof. A polynomial $Q(x, y)$ with $(1, k-1)$-weighted degree at most l can be of the form

$$Q(x, y) = \sum_{j=0}^{\lfloor \frac{l}{k-1} \rfloor} \sum_{i=0}^{l-(k-1)j} q_{ij} x^i y^j.$$

Given that $Q(\alpha_i, \beta_i) = 0$ for all $i = 1, \cdots, n$, we have a linear system of n equations in the unknowns q_{ij}.

For exixtence of a nonzero solution to $Q(x, y)$, we study the rank of the coefficient matrix of the linear system. Suppose the number of unknowns is Δ, by the number of monomials in

$Q(x, y)$, we have

$$\begin{aligned}
\Delta &= \sum_{j=0}^{\lfloor \frac{l}{k-1} \rfloor} \sum_{i=0}^{l-(k-1)j} 1 \\
&= \sum_{j=0}^{\lfloor \frac{l}{k-1} \rfloor} (l-(k-1)j+1) \\
&= \left(\left\lfloor \frac{l}{k-1} \right\rfloor + 1 \right)(l+1) - \left(\left\lfloor \frac{l}{k-1} \right\rfloor + 1 \right) \left\lfloor \frac{l}{k-1} \right\rfloor \frac{(k-1)}{2} \\
&= \left(\left\lfloor \frac{l}{k-1} \right\rfloor + 1 \right) \left(l+1 - \frac{(k-1)}{2} \left\lfloor \frac{l}{k-1} \right\rfloor \right) \\
&\geq \left(\left\lfloor \frac{l}{k-1} \right\rfloor + 1 \right) \left(l+1 - \frac{l}{2} \right) \\
&\geq \frac{l}{k-1} \left(\frac{l+2}{2} \right).
\end{aligned}$$

Since the rank is at most the number of constraints in the linear system, $n\binom{m+1}{2}$, the result follows. □

Example 4.1.6. Consider the $[6, 3, 4]_7$ Reed-Solomon code. Suppose the received vector is $(0, 1, 4, 1, 3, 4)$. This means we are given the points

$$(0,0), (1,1), (2,4), (3,1), (4,3), (5,4) \in \mathbb{F}_7^2,$$

the number of agreements $t > \sqrt{n(k-1)}$ so we set $t = 4$, we can compute the multiplicity parameter, m and the maximum list size l as defined in [3]:

$$\begin{aligned}
m &:= 1 + \left\lfloor \frac{(k-1)n + \sqrt{(k-1)^2 n^2 + 4(t^2 - (k-1)n)}}{4(t^2 - (k-1)n)} \right\rfloor, \\
l &:= mt - 1,
\end{aligned}$$

in which case $m = 2$ and $l = 7$. We kill every monomial of the following polynomial whose $(1, 1)$-weighted degree is less than $m = 2$.

$$Q(x,y) = \sum_{j=0}^{\lfloor \frac{l}{k-1} \rfloor} \sum_{i=0}^{l-(k-1)j} q_{ij} x^i y^j = \sum_{j=0}^{4} \sum_{i=0}^{4-j} q_{ij} x^i y^j$$

If $Q(x, y)$ is zero when evaluated at each of the given points, we attain a homogeneous linear equation to solve for some variables q_{ij}. One such polynomial from the linear system is

$$Q(x,y) = -2x^7 - 2x^6 + 3x^5 y - 3x^5 - 3x^4 y + 2x^3 y^2 + 3x^3 y - 2x^2 y^2 - 3xy^3 + x^3 - xy.$$

This completes the interpolation step of the algorithm.

For the factorisation, we want to get all polynomials of degree at most $k-1$ (i.e. 2) in x and check that at least $t=4$ of the given points form roots to them.

We get the following factorisation over $\mathbb{F}_7$

$$Q(x,y) = -3x(-x^2+y)(x^2-2x+y+3)(-3x^2-2x+y-3)$$

from which we have $-x^2+y=0$ for the points $(0,0),(1,1),(2,4),(5,4)$ with coefficients in $\mathbb{F}_7$, $x^2-2x+y+3=0$ only for the three points $(2,4),(3,1),(4,3)$ with coefficients in $\mathbb{F}_7$, and $-3x^2-2x+y-3=0$ for the points $(1,1),(3,1),(4,3),(5,4)$, with coefficients in $\mathbb{F}_7$. Our list is

$$\mathcal{L} = \{x^2, 3x^2+2x+3\}.$$

From Example 4.1.4, we know these listed polynomials identify with codewords in the following way:

- $y = 3x^2+2x+3$ is just 3 times row 3 plus 2 times row 2 plus 3 times row 1 of the generator matrix of our $[6,3,4]_7$ Reed-Solomon code. In which case, $(\boxed{3},1,\boxed{5},1,3,4)$ must be the sent codeword;
- $y = x^2$ is just row 3 of the generator matrix of our $[6,3,4]_7$ Reed-Solomon code - $(0,1,4,\boxed{2},\boxed{2},4)$ must be the sent codeword. See that the coordinates 4 and 5 were those corrupted during transmission.

The possible messages transmitted are $(3,1,5,1,3,4)$ and $(0,1,4,2,2,4)$.

4.2 A Univariate Polynomial Representation

The interpolation step constructs a bivariate polynomial, Equation (4.1.1). It is easier to work with a univariate polynomial, instead, so we describe a general way to put down a univariate representation for any bivariate polynomial. From Equation (4.1.1), we define the set of all monomials as

$$M_{x,y} := \left\{x^i y^j : i,j \geq 0\right\}. \tag{4.2.1}$$

From this definition, we denote $\mathbb{Z}_{\geq 0}$ as the set of nonnegative integers and construct the map

$$\begin{aligned} \varphi : M_{x,y} &\longrightarrow \mathbb{Z}^2_{\geq 0} \\ \varphi(x^i y^j) &\longmapsto (i,j). \end{aligned} \tag{4.2.2}$$

The map φ in Equation (4.2.2) is a bijection of sets.

Definition 4.2.1 (Monomial ordering [18])**.** A monomial order is a total order '$<$' on the set of monomials $M_{x,y}$ such that

- ⋆ if $\lambda_1 \leq \delta_1$, $\lambda_2 \leq \delta_2$, then $(\lambda_1,\lambda_2) \leq (\delta_1,\delta_2)$;

⋆ if $\lambda \leq \delta$ and $\lambda, \delta, \gamma \in \mathbb{Z}_{\geq 0}$, then $\lambda + \gamma \leq \delta + \gamma$.

Definition 4.2.2 (Weighted degree [18])**.** For u, v nonnegative integers, the (u, v)-weighted degree of a monomial $x^i y^j$ is

$$\deg_{(u,v)} x^i y^j = ui + vj. \tag{4.2.3}$$

Definition 4.2.3 ((u, v)-lexicographic order [18])**.** For two monomials $x^{i_1} y^{j_1} < x^{i_2} y^{j_2}$, implies that $ui_1 + vj_1 < ui_2 + vj_2$, or $ui_1 + vj_1 = ui_2 + vj_2$ and $i_1 < i_2$.

Definition 4.2.4 ((u, v)-reverse lexicographic order [18])**.** For two monomials $x^{i_1} y^{j_1} < x^{i_2} y^{j_2}$, implies that $ui_1 + vj_1 > ui_2 + vj_2$, or $ui_1 + vj_1 = ui_2 + vj_2$ and $i_1 > i_2$.

Example 4.2.5. Consider the polynomial

$$Q(x, y) = -2x^7 - 2x^6 + 3x^5 y - 3x^5 - 3x^4 y + 2x^3 y^2 + 3x^3 y - 2x^2 y^2 - 3xy^3 + x^3 - xy.$$

A $(1, 2)$-weighted degree lexicographic ordering of the monomials of $Q(x, y)$ is

$$xy < x^3 < x^3 y < x^5 < x^2 y^2 < x^4 y < x^6 < xy^3 < x^3 y^2 < x^5 y < x^7.$$

If we order the set $M_{x,y}$ considering Equation (4.2.3) in a lexicographic or reverse lexicographic order, we essentially have a univariate representation

$$\varphi_0(x, y) < \varphi_1(x, y) < \varphi_2(x, y) < \cdots < \varphi_l(x, y)$$

for some finite number of $\varphi_i(x, y) \neq 0$ in the variable y. That is to say, the bivariate polynomial can be written as

$$Q(x, y) = \sum_{j=0}^{l} c_j \varphi_j(x, y) \tag{4.2.4}$$

where c_j is an element of the field. This is important because it reduces the problem at the factorisation step to a root finding problem.

Example 4.2.6. Consider the $[7, 4, 4]_7$ Reed-Solomon code. Given the points

$$(0, 0), (1, 3), (2, 4), (3, 1), (4, 6), (5, 4), (6, 1) \in \mathbb{F}_7^2,$$

the number of agreements $t > \sqrt{n(k-1)}$ so we set $t = 5$, we compute $m = 2$ and $l = 9$ from the formulas in Example 4.1.6, and kill every monomial of the following polynomial whose $(1, 1)$-weighted degree is less than $m = 2$:

$$Q(x, y) = \sum_{j=0}^{3} \sum_{i=0}^{9-3j} q_{ij} x^i y^j.$$

If $Q(x, y)$ is zero when evaluated at each of the given points, we attain a homogeneous linear equation to solve for some variables q_{ij}. One such polynomial from the linear system is

$$\begin{aligned} Q(x, y) = &- 2x^9 - 2x^8 + 2x^6 y - 2x^6 - 3x^5 y - 2x^5 + x^4 y - 3x^3 y^2 - 2x^4 - x^3 y + 2x^2 y^2 - 3x^3 \\ &+ 3y^3 + x^2 + xy - 2y^2. \end{aligned}$$

This completes the interpolation step of the algorithm.

Instead of factorisation, we write a univariate representation of $Q(x, y)$ by using a $(1, 1)$-weighted degree lexicographic ordering of $Q(x, y)$. This is just

$$\begin{aligned} Q(x,y) =& (-2x^9 - 2x^8 - 2x^6 - 2x^5 - 2x^4 - 3x^3 + x^2) + (2x^6 - 3x^5 + x^4 - x^3 + x)y \\ &+ (-3x^3 + 2x^2 - 2)y^2 + 3y^3. \end{aligned}$$

We solve this polynomial in y to get the roots $y = x^3 + 2x^2 + 3x + 3$, $y = -2x^3 + 2x^2 + 3x$ and $y = 2x^3 + x$, all polynomials of at most degree 3. Now we check that at least $t = 5$ of the given points form roots to these three polynomials in x over $\mathbb{F}_7$. We have agreements for $y = x^3 + 2x^2 + 3x + 3$ at the points $(0, 0), (1, 3), (2, 4), (3, 1), (4, 6)$, for $y = -2x^3 + 2x^2 + 3x$ at the points $(0, 0), (1, 3), (3, 1), (5, 4), (6, 1)$ and for $y = 2x^3 + x$ at the points $(2, 4), (3, 1), (4, 6), (5, 4), (6, 1)$ so we list all three

$$\mathcal{L} = \{x^3 + 2x^2 + 3x + 3, -2x^3 + 2x^2 + 3x, 2x^3 + x\}.$$

The possible messages transmitted are $(3, 2, 4, 1, 6, 4, 1)$, $(0, 3, 5, 1, 0, 4, 1)$ and $(0, 4, 6, 4, 2, 1, 3)$.

4.3 Hasse Derivatives

Given an interpolation polynomial (any polynomial) Q, a well-known technique of finding its singularities is studying its partial derivatives. We shall use Hasse derivatives, instead, to study the singularities of Q (points where Q intersects itself) simply because partial derivatives of polynomials over fields of small characteristic are not well behaved.

Definition 4.3.1 ((r, s)-Hasse derivative [20])**.** The (r, s) Hasse derivative of a polynomial $Q(x, y)$, denoted $D_{r,s}Q(x, y)$, for any integer pair $r \geq 0$, $s \geq 0$ is

$$D_{r,s}Q(x,y) = \sum_{i,j} \binom{i}{r}\binom{j}{s} q_{i,j} x^{i-r} y^{j-s} \tag{4.3.1}$$

where $q_{i,j}$ is the coefficient of $x^i y^j$ in $Q(x, y)$.

In the Guruswami-Sudan algorithm, studying singularities by exploring partial derivatives was avoided for a more suitable technique. The technique was to shift a coordinate system to a chosen point (x_i, y_i) as a new origin, and then insist that the non-zero coefficients be of high degree. The shifting for $Q(x, y)$, where $\alpha, \beta \in \mathbb{F}_q$ is

$$Q_{\alpha,\beta}(x,y) := Q(x + \alpha, y + \beta). \tag{4.3.2}$$

This technique is essentially applying Hasse derivatives to the polynomial $Q(x, y)$ since

$$D_{r,s}Q(\alpha,\beta) = \text{coefficient}_{x^r y^s} Q(x + \alpha, y + \beta).$$

This is the statement in Theorem 4.3.2. The idea of applying Hasse derivatives on interpolation polynomials makes it somewhat easier to study their singularities with multiplicities. This was discussed as a product in Lemma 2 of [20].

The bivariate polynomial Q is said to have a zero of multiplicity m at (α, β) if the shift (Equation (4.3.2)) has a zero of multiplicity m at $(0, 0)$. In other words, every monomial in Q with $\deg_{(1,1)} x^i y^j < m$ has zero coefficient. The primary significance of Hasse derivatives is to help express the shift $Q(x + \alpha, y + \beta)$ as a bivariate in x and y, which can be given a univariate representation using Equation (4.2.4) for easy factorization.

The following theorem was given in the work of McEliece and we present the exact proof given there due to its elegance.

Theorem 4.3.2 (See Theorem 4.4 in [18]). *Let*

$$Q(x, y) = \sum_{i,j} q_{i,j} x^i y^j \in \mathbb{F}_q[x, y].$$

For any $(\alpha, \beta) \in \mathbb{F}_q^2$, *the shift*

$$Q(x + \alpha, y + \beta) = \sum_{r,s} D_{r,s} Q(\alpha, \beta) x^r y^s.$$

Proof. The idea is to use binomial expansion such that the shift

$$\begin{aligned} Q(x + \alpha, y + \beta) &= \sum_{i,j} q_{i,j} (x + \alpha)^i (y + \beta)^j \\ &= \sum_{i,j} q_{i,j} \left(\sum_r \binom{i}{r} x^r \alpha^{i-r} \right) \left(\sum_s \binom{j}{s} y^s \beta^{j-s} \right) \\ &= \sum_{r,s} x^r y^s \left(\sum_{i,j} \binom{i}{r} \binom{j}{s} q_{i,j} \alpha^{i-r} \beta^{j-s} \right) \\ &= \sum_{r,s} D_{r,s} Q(\alpha, \beta) x^r y^s. \end{aligned}$$

□

Corollary 4.3.3 ([18]). We have

$$Q(x, y) = \sum_{r,s} D_{r,s} Q(\alpha, \beta) (x - \alpha)^r (y - \beta)^s.$$

A second look at the factorization step suggests that using Corollary 4.3.3, the list in Equation (4.1.2) is just

$$\mathcal{L} = \{ f(x) \in \mathbb{F}_q[x] : (y - f(x)) \mid Q(x, y) \}.$$

Our goal here is to show the significance of Hasse derivatives in solving the Guruswami-Sudan algorithm.

4.4 Uses of Reed-Solomon Codes

Reed-Solomon codes have found use in many systems since the quest for optimality is prioritized in engineering and telecommunication. We outline a few well-known uses here.

- Consumer technologies like Compact Disc (CD), Digital Optical Disc (DVD), Blu-ray Discs, QR Codes;
- Data transmission technologies like Digital Scriber Line (DSL) and WiMAX, a family of wireless communications standards;
- Broadcast systems like Digital Video Broadcasting (DVB), and Advance Television Systems Committee Standards (ATSCS), standards for digital transmission over terrestrial cable and satellite networks;
- Storage systems like RAID 6, which are configurations for stripping, mirroring or creating parity to actually create large reliable data stores on hard disk drives.

See [4] among other sources for more on this.

In this chapter, we actually built some good foundation for understanding both the Guruswami-Sudan problem and algorithm. In Chapter 5, we will present some improvement techniques for the Guruswami-Sudan algorithm.

5. Improved Complexity

In Chapter 4, we built some tools for the improvement process. Here, we present some complexity improvement techniques that were based on the Guruswami-Sudan problem and algorithm. We discuss reconstruction in Section 5.2 and transformation in Section 5.3.

5.1 Error-rate as a Function of Message Rate

Let $\mathcal{C}$ be an $[n, k, d]_q$ error-correcting code (i.e. $\mathcal{C}$ has codeword length n, dimension k and minimum distance d over a q-ary alphabet $\mathbb{F}_q$). This means the map

$$\mathcal{C} : \mathbb{F}_q^k \longrightarrow \mathbb{F}_q^n$$

such that if $c_1, c_2 \in \mathcal{C}$ then c_1 and c_2 differ in at least d coordinates. For a Reed-Solomon code, $d \geq n - k + 1$ with $k < n \leq q$. We use this construction to study the error-rate dependence on the message rate in decoding the Reed-Solomon code $\mathcal{C}$.

Let e be the number of distinct coordinates between some two vectors in $\mathbb{F}_q^n$ where one is transmitted (a codeword) and another, a message is received. The number of alterations that can be corrected - the error in the transmission - by Theorem 3.2.2, is $2e < d$. The error-rate is defined as $\varepsilon := \frac{e}{n}$, while the message rate is $\kappa := \frac{k}{n}$. We study the error-rate dependence on the message rate in two cases. These are: (1) when both the error-rate and the message rate are fixed, and (2) when the error-rate is not fixed.

Basically, both rates are fixed when the number of errors is within the unique-decoding bound. A unique decoding bound μ is a bound for an error correcting code such that for a ball of radius μ centered at a received message, there can be only one codeword within the ball. A powerful algorithm by Peterson [21] constructed that for a decoding radius

$$e < \frac{n-k}{2},$$

we can decode without any complications. It can be observed that the error-rate is

$$\varepsilon = \frac{1-\kappa}{2}$$

where the message rate is fixed (i.e. $\kappa := \frac{k}{n}$). Peterson's algorithm [21] provides a suitable alternative to very laborious techniques of computing the error-rate which are comparing a received word with every other codeword, or compare a received word with those codewords in its vicinity. These techniques work but require a lot of time. See Berlekamp's book [19] for more information on the techniques. However, this algorithm is also rendered inefficient when the decoding radius, e is beyond the unique-decoding bound.

It might happen that a list of codewords are found within a certain ball of radius

$$e \geq \frac{n-k}{2}.$$

Hence, unique-decodability might not be achieved here. The problem of decoding correctly in this case is interesting to researchers. Therefore, we shall study its error-rate dependence on the message rate. From the list decoding algorithm given by Guruswami-Sudan [3], a polynomial-time algorithm with the capability of decoding up to (for an $[n, k+1, d]_q$ code)

$$e = \lfloor n - \sqrt{nk} \rfloor \tag{5.1.1}$$

radius is acquired. Some later authors improved on this algorithm and hence arrived at some improved complexities as a result of some re-encoding, and some reformulation techniques. However, we are basing our analysis on this algorithm because it is very general and the most standard, and does not camouflage the situation to get better result as some other methods present. We observe that

$$\varepsilon = 1 - \sqrt{\kappa} = \frac{1-\kappa}{1+\sqrt{\kappa}} \geq \frac{1-\kappa}{2}. \tag{5.1.2}$$

One of the reconstructions based on the Guruswami-Sudan algorithm was done in [4] and the authors argued that the decoding radius could be larger than that given by Guruswami-Sudan algorithm in [3]. However, this was as a result of making some improvements to the codewords (reconstruction) prior to the application of the Guruswami-Sudan algorithm. We want to note that the error-rate and message rate dependence was discussed in [3] and we expand on it here.

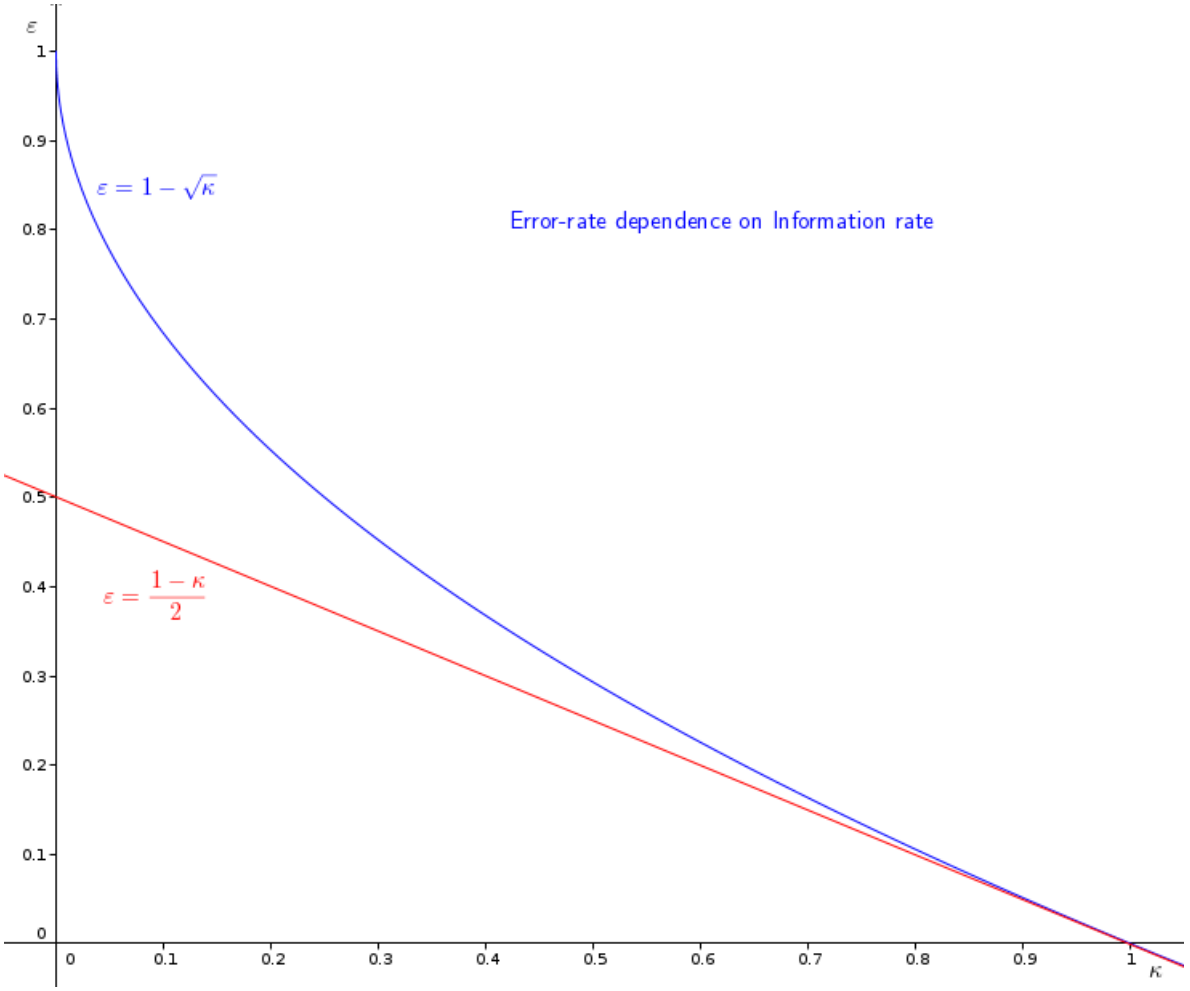

Figure 5.1: Error-rate dependence on information rate

5.2 Polynomial Reconstruction

Let $\mathcal{C}$ be an $[n, k, d]_q$ Reed-Solomon code. We know from Section 5.1 that the Guruswami-Sudan algorithm corrects within a wider Hamming ball than classical decoding algorithms. When we extend its decoding radius, some received words that have errors might not be correcrtly decoded. Since the Guruswami-Sudan algorithm is our reference point (standard algorithm), it will be interesting to investigate possible ways to improve its error-correction capabilities. Muralidhara and Sen [4] presented a construction that discusses one way of doing this. Note that the Guruswami-Sudan problem is the following: given n distinct points $(x_1, x_2, \cdots, x_n)$ in $\mathbb{F}_q$ and another n points $(y_1, y_2, \cdots, y_n)$ in $\mathbb{F}_q$, find polynomials $f(x)$ of degree at most $k-1$ that agree at at least t points, i.e.

$$\Gamma := \{f(x) \in \mathbb{F}_q[x] | \deg f(x) \leq k-1, f(x_i) = y_i \text{ for at least } t \text{ points}\}. \tag{5.2.1}$$

Finding ALL polynomials that agree at t points is equivalent to listing ALL codewords within a Hamming ball of radius $n - t$.

Let $k < \lambda n$ for $0 < \lambda < 1$. Considering the Guruswami-Sudan problem, define a polynomial

$$f_j(x_i) := \frac{y_i - y_j}{x_i - x_j}, i \neq j. \tag{5.2.2}$$

Denote $\mathcal{C}$ an $[n, k, d]_q$ Reed-Solomon code having codewords agreeing at at least t points as an (n, k, t) instance of the Guruswami-Sudan problem. If we let $j = 1$, then Equation (5.2.2) becomes

$$f_1(x_i) := \frac{y_i - y_1}{x_i - x_1}, i = 2, \cdots, n \tag{5.2.3}$$

with $n - 1$ points

$$\left(\frac{y_2 - y_1}{x_2 - x_1}, \frac{y_3 - y_1}{x_3 - x_1}, \cdots, \frac{y_n - y_1}{x_n - x_1} \right).$$

The goal is to reduce the degree from k to $k - 1$. If $f(x_1) = y_1$ and deg $f(x) \leq k - 1$, then deg $f_1(x) \leq k - 2$ since the n-distinct input points $(x_1, x_2, \cdots, x_n)$ is reduced to an $(n - 1)$-distinct input points $(x_2, \cdots, x_n)$ (see Theorem 5.2.1). At this stage, we have reduced to an $(n-1, k-1, t-1)$ instance of the problem. In [4], the authors attained a very striking result regarding the list size of the codewords after the Guruswami-Sudan algorithm is applied to the $(n - 1, k - 1, t - 1)$ instance.

Theorem 5.2.1 (See Lemma 2.1 in [4]). *There exists a polynomial $f(x)$ such that deg $f(x) \leq k - 1$ and $f(x_i) = y_i$ for at least t points and $f(x_1) = y_1$ iff there is a polynomial $f_1(x)$ of degree at most $k - 2$ such that*

$$f_1(x_i) := \frac{y_i - y_1}{x_i - x_1}, i = 2, \cdots, n$$

for at least $t - 1$ points.

Proof. ($\Longrightarrow$) By Euclidean algorithm, we let

$$\frac{f(x)}{x - x_1} = f_1(x) + \frac{f(x_1)}{x - x_1}.$$

This implies that

$$f(x) = f_1(x)(x - x_1) + y_1.$$

It is straight forward to see that indeed deg $f_1(x) \leq k - 2$.

($\Longleftarrow$) We know that $f_1(x)(x - x_1)$ is a polynomial, so define

$$f(x) := f_1(x)(x - x_1) + y_1.$$

We see that $f(x_1) = y_1$ and deg $f(x) \leq k - 1$. □

If we consider Equation (5.2.1) and define

$$\Gamma_j := \left\{ f_j(x) \in \mathbb{F}_q[x] | \text{deg } f_j(x) \leq k - 2, f_j(x_i) = \frac{y_i - y_j}{x_i - x_j}, i \neq j \text{ for at least } t - 1 \text{ points} \right\}, \tag{5.2.4}$$

then Equation (5.2.1) can be written as

$$\Gamma = \cup_{j=1}^{n} \{f_j(x)(x - x_j) + y_j | f_j(x) \in \Gamma_j\}. \tag{5.2.5}$$

Since for all $f \in \Gamma$, $f(x_i) = y_i$ for at least t points and can be recovered from Γ_j by the reduction process explained above.

For an (n, k, t) Guruswami-Sudan problem, we let $t > \sqrt{nk}$ in the decoding radius given in Equation (5.1.1). Similarly, for an $(n-1, k-1, t-1)$ we need $t - 1 > \sqrt{(n-1)(k-1)}$ for efficient decoding. If $t - 1 < \sqrt{(n-1)(k-1)}$, we do further reduction of the remaining $(n-1)$-distinct input points $(x_2, \cdots, x_n) \in \mathbb{F}_q$ for some r times to get an $(n-r, k-r, t-r)$ instance of the problem, with

$$t - r > \sqrt{(n-r)(k-r)}. \tag{5.2.6}$$

For n distinct points $(x_1, x_2, \cdots, x_n) \in \mathbb{F}_q$, $\exists\ n^r$ choices in constructing an $(n-r, k-r, t-r)$ instance from an (n, k, t) instance. Therefore, applying the Guruswami-Sudan algorithm to this reduced state n^r times is equivalent to solving the original problem. This method of polynomial reconstruction is optimal in the sense that no information is lost and the list size of codewords generated by the Guruswami-Sudan algorithm is maintained. In [4], the authors gave an upper bound for r such that Equation (5.2.6) holds.

Theorem 5.2.2 (See Theorem 2.3 in [4]). *Let $k < \alpha n$, $0 < \alpha < 1$ and $\sqrt{nk} - 1 < t \leq \sqrt{nk}$. Let*

$$\Gamma = \{f(x) \in \mathbb{F}_q[x] | deg\ f(x) \leq k-1, f(x_i) = y_i\ for\ at\ least\ t\ points\}.$$

Then the

1. *number of polynomials, $|\Gamma| \leq O\left(n^{\frac{2\sqrt{\alpha}}{(1-\sqrt{\alpha})^2}+3}\right)$;*
2. *elements in Γ can be listed in $O\left(n^{\frac{2\sqrt{\alpha}}{(1-\sqrt{\alpha})^2}+11}\right)$ time.*

Proof. Suppose $k < \alpha n$. We apply the reduction $r = \frac{2\sqrt{\alpha}}{(1-\sqrt{\alpha})^2} + 1$ times. By easy manipulation of our assumed fact, we can write

$$r = \frac{2\sqrt{\alpha}}{(1-\sqrt{\alpha})^2} + 1 > \frac{2\sqrt{\alpha} + \frac{1}{n}}{(1-\sqrt{\alpha})^2} > \frac{2\sqrt{\frac{k}{n}} + \frac{1}{n}}{\left(1 - \sqrt{\frac{k}{n}}\right)^2}.$$

This implies that

$$\frac{1}{r} < \frac{n + k - 2\sqrt{nk}}{2\sqrt{nk} - 1}.$$

Consider $\sqrt{nk}-1 < t \leq \sqrt{nk}$. We have $-2rt > 2\sqrt{nk}-1-r(n+k)$. Since $2\sqrt{nk}-1 > nk-t^2$,

$$\begin{aligned} -2rt > nk - t^2 - r(n+k) &\iff t^2 - 2rt - r > nk - r(n+k) - r > nk - r(n+k) - r^2 \\ &\iff (t-r)^2 > (n-r)(k-r) \\ &\iff (t-r) > \sqrt{(n-r)(k-r)}. \end{aligned}$$

The reconstruction problem is solved for $\sqrt{nk} - 1 < t \leq \sqrt{nk}$ by applying the Guruswami-Sudan algorithm $O\left(n^{\frac{2\sqrt{\alpha}}{(1-\sqrt{\alpha})^2}+1}\right)$ times. In [22] and [23], the Guruswami-Sudan algorithm algorithm is shown to take $O(n^{10})$ time and returns a list of size $O(n^2)$. These two complexities are contained in our claim, so the theorem is proved. □

For a Hamming radius $e = \lfloor n - \sqrt{nk} \rfloor$, the Guruswami-Sudan algorithm guarantees a successful decoding process. It is presented in [4] that even for a radius of

$$\lfloor n - \sqrt{nk} \rfloor + c, c > 0, \tag{5.2.7}$$

(which is greater than the Guruswami-Sudan radius) the reduction process explained above can complement the Guruswami-Sudan algorithm for unique-decoding resulting in improved complexities. The error-rate in this case,

$$\varepsilon = 1 - \sqrt{\kappa} + \frac{c}{n}, \tag{5.2.8}$$

is actually a negligible difference from the error-rate attained by the Guruswami-Sudan algorithm given as Equation (4.3.2). However, it is typical of divide-and-conquer algorithms that a minor factor of improvement could result in a significant improvement in the total running time. The reduction procedure is indeed a divide-and-conquer approach (an approach that breaks a problem into sub-problems, recursively solve the sub-problems and then combine the answers).

The following theorem is a generalisation of Theorem 5.2.2 so its proof can be done by induction on c, with base case $c = 1$, and techniques used in the proof of Theorem 5.2.2.

Theorem 5.2.3 (See Theorem 2.4 in [4]). *Let $k < \alpha n$, $0 < \alpha < 1$ and $\sqrt{nk} - c < t \leq \sqrt{nk} - c + 1$ for $c > 0$. Let*

$$\Gamma = \{f(x) \in \mathbb{F}_q[x] | deg\ f(x) \leq k-1, f(x_i) = y_i \text{ for at least } t \text{ points}\}.$$

Then the

1. *number of polynomials, $|\Gamma| \leq O\left(n^{c\frac{2\sqrt{\alpha}}{(1-\sqrt{\alpha})^2}+c+2}\right)$;*
2. *polynomials in Γ can be listed in $O\left(n^{\frac{2\sqrt{\alpha}}{(1-\sqrt{\alpha})^2}+c+10}\right)$ time.*

5.3 Alternative Basis Transformation

5.3.1 Lagrange Interpolation.

Theorem 5.3.2 ([15]). *Given n distinct elements $(x_1, x_2, \cdots, x_n) \in \mathbb{F}_q$ and another n elements $(y_1, y_2, \cdots, y_n) \in \mathbb{F}_q$, there exists exactly one polynomial $R(x)$ of degree less than n with coefficients in $\mathbb{F}_q$ such that*

$$R(x_i) = y_i, i = 1, 2, \cdots, n.$$

Proof. First, we prove that this polynomial exists, and then we show that it is unique. Now, define some polynomial

$$R(x) := \sum_{i=1}^{n} y_i L_i(x)$$

where

$$L_i(x) := \prod_{j=1, j\neq i}^{k} \frac{x - x_j}{x_i - x_j}.$$

This shows that such a polynomial exists.

For the uniqueness, assume that $R_1(x)$ and $R_2(x)$ both satisfy the conditions in the theorem. Then $R_1(x) - R_2(x)$ has n distinct roots. Then by Theorem 4.1.2, $R_1(x) - R_2(x)$ is a zero polynomial. Therefore, there exists such a polynomial and it is unique. □

5.3.3 Syndrome-Based Decoding. As before, let e be the number of errors that occur on a transmitted codeword $x \in C$ such that $y \in \mathbb{F}_q^n$ is received. We denote a vector E as the error vector where the coordinates of E hold information about the locations of alterations. Thus, $E = (E_1, E_2, \cdots, E_n)$ with $E_i \in \mathbb{F}_q$ and

$$E_i = \begin{cases} 1 & \text{if there is an error at coordinate } i, \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathcal{J}$ be the set of error locations in a received vector y (i.e. all $E_i = 1$). We have stated in Section 4.1 that in classical decoding we have $e = \left\lfloor \frac{n-k}{2} \right\rfloor$, and an $[n, k, d]_q$ code can correct any error pattern E if and only if $e \geq \mid \mathcal{J} \mid$.

In the book by Ron Roth [24], we see that for syndrome coefficients $S_0, S_1, \cdots, S_{n-k-1}$ and E the error vector, the syndrome polynomial $S(x)$ is defined as

$$S(x) := \sum_{i=0}^{n-k-1} S_i x^i \equiv \sum_{j=1}^{n} \frac{E_j v_j}{1 - \delta_j x} \bmod x^{n-k}$$

where $\delta_j, v_j, j = 1, \cdots, n$ are some multipliers.

Definition 5.3.4 ([24]). In syndrome decoding, the error-locator polynomial is defined as

$$\Lambda(x) := \prod_{j \in \mathcal{J}} (1 - \delta_j x)$$

and the error-evaluator polynomial is defined as

$$\Omega(x) := \sum_{j \in \mathcal{J}} E_j v_j \prod_{i \in \mathcal{J} \setminus \{j\}} (1 - \delta_i x).$$

From Definition 5.3.4, we have $\Lambda(0) \neq 0$ so the (well-defined) fraction

$$\frac{\Omega(x)}{\Lambda(x)} = \frac{\sum_{j \in \mathcal{J}} E_j v_j}{\sum_{j \in \mathcal{J}} (1 - \delta_j x)} \equiv \sum_{j \in \mathcal{J}} \frac{E_j v_j}{1 - \delta_j x} \bmod x^{n-k}.$$

Since the above fraction is well-defined, we have a Key Equation that relates the error-locator and the error-evaluator polynomials as

$$\frac{\Omega(x)}{\Lambda(x)} \equiv S(x) \bmod x^{n-k}. \tag{5.3.1}$$

In syndrome decoding, the decoder basically computes $S(x)$ from the received word y, solves the Key Equation (i.e. Equation (5.3.1)) for the error-locator polynomial $\Lambda(x)$ to determine its roots, and finally compute the error-evaluator $\Omega(x)$ to determine the error values. We have discussed a general case of simple syndrome decoding in Example 3.3.6.

We will use the basic idea of syndrome-based decoding of Reed-Solomon codes to reformulate the Guruswami-Sudan problem (given in Section 4.1) in terms of modules over a univariate polynomial ring (in other words, Key Equations).

5.3.5 Modules over $\mathbb{F}_q[x]$. Consider the Guruswami-Sudan problem (we restate it again) - given n distinct points $(x_1, x_2, \cdots, x_n)$ in $\mathbb{F}_q$ and another n points $(y_1, y_2, \cdots, y_n)$ in $\mathbb{F}_q$, find polynomials $f(x)$ of degree at most $k-1$ that agree at at least t points. Our goal is to reformulate this problem in terms of modules over a univariate polynomial ring, $\mathbb{F}_q[x]$.

Definition 5.3.6 (Module [25]). Let M be a ring. A (left) M-module is an additive Abelian group A together with a function $M \times A \longmapsto A$ such that for all $m_1, m_2 \in M$ and $a, b \in A$:

1. $m_1(a + b) = m_1 a + m_1 b$,
2. $(m_1 + m_2)a = m_1 a + m_2 a$,
3. $m_1(m_2 a) = (m_1 m_2)a$.

Let $\alpha = (\alpha_1, \cdots, \alpha_n)$ be some transmitted codeword and $\beta = (\beta_1, \cdots, \beta_n)$ be the received word. The Guruswami-Sudan algorithm has order of multiplicity parameter m (which is 1

in Sudan [2]) for the n points $(\alpha_1, \beta_1), (\alpha_2, \beta_2), \cdots, (\alpha_n, \beta_n)$. Let l be the length of the list in Equation (4.1.2) and τ be the maximum radius decoding. The bivariate polynomial

$$Q(x, y) = \sum_{t=0}^{l} Q_t(x) y^t \in \mathbb{F}_q[x, y]$$

is the polynomial we look for in the interpolation step which satisfies some conditions including the $(1, k-1)$-weighted degree must be as small as possible. This means, for any monomial $x^i y^j$ in $Q(x, y)$, $i + (k-1)j < m(n-\tau)$ (this is because the length of the list as in Example 4.1.6 is $l < mt = m(n-\tau)$). By our construction of $Q(x, y)$, this weighted degree condition has that

$$\deg Q_t(x) < m(n-\tau) - (k-1)t, \text{ for } t = 0, 1, \cdots, l.$$

Hence, we define

$$N_t := m(n-\tau) - (k-1)t, \text{ for } t = 0, 1, \cdots, l.$$

All we need to know about the multiplicity parameter is that for any monomial $x^i y^j$ in $Q(x, y)$, the coefficient q_{ij} of $x^i y^j$ is zero whenever $i + j < m$.

Let $R(x)$ be the Lagrange interpolation polynomial such that $R(\alpha_i) = \beta_i$ and define some polynomial $G(x) := \prod_{i=1}^{n}(x - \alpha_i)$. We know from Corollary 4.3.3 that

$$Q(x, y) = \sum_{r,s} D_{r,s} Q(\alpha, \beta)(x-\alpha)^r (y-\beta)^s = \sum_{r,s} D_{r,s} Q_t(\alpha) \beta^t (x-\alpha)^r (y-\beta)^s.$$

Therefore, having $r = 0$, $0 \le t \le l$, and $0 \le s < m$ gives that

$$Q(x, y) = \sum_{s=0}^{l} D_{0,s} Q_t(\alpha) \beta^t (y-\beta)^s. \tag{5.3.2}$$

From Equation (5.3.2), we see that $Q(x, y)$ has a multiplicity at least m at (α_i, β_i) if and only if $(x-\alpha_i)^{m-s}$ divides $D_{0,s} Q_t(\alpha) \beta^t$ for all s, t with $0 \le s < m$ and $t = 0, 1, \cdots, l$. Below, we prove a result which is fundamental in the reformulation process. Similar results were presented in [11] and [12].

Proposition 5.3.7. A bivariate polynomial $Q(x, y) \in \mathbb{F}_q[x, y]$ has multiplicity at least m at every (α_i, β_i) if and only if $G(x)^{m-s} \mid D_{0,s} Q(x, R(x))$ for all s with $0 \le s < m$.

We will prove that the multiplicity condition implies each factor $(x-\alpha_i)^{m-s}$ divides $D_{0,s} Q(x, R(x))$ and by the uniqueness in the factors for all i, the product $G(x)^{m-s}$ divides it. For the converse, remainder theorem and Hasse derivatives will be helpful.

Proof. Let $(\alpha_i, \beta_i) \in \mathbb{F}_q^2$, $R(x) \in \mathbb{F}_q[x]$ such that $R(\alpha_i) = \beta_i$. Suppose $Q(x, y)$ has multiplicity at least m at (α_i, β_i). Equation (4.2.4) gives a univariate representation such that

$$D_{0,s} Q(x, R(x)) = \sum_{i \ge m-s} D_{0,s} c_i \varphi_i(x, R(x)). \tag{5.3.3}$$

We shift to the origin - assume $(\alpha_i, \beta_i) = (0, 0)$. This means $R(0) = 0$, so $x \mid R(x)$. Consider $s < m$ where we have each polynomial $D_{0,s}\varphi_i(x, y)$ with no terms of degree less than $m - s$. Since each $D_{0,s}\varphi_i(x, y)$ have have only terms with degrees greater than or equal to $m - s$, and $x \mid R(x)$, then

$$x^{m-s} \mid D_{0,s}c_i\varphi_i(x, R(x)) \text{ for all } s \text{ with } 0 \leq s < m.$$

Shifting back, it follows that

$$(x - \alpha_i)^{m-s} \mid D_{0,s}c_i\varphi_i(x, R(x)) \text{ for all } s, i \text{ with } 0 \leq s < m \text{ and } 1 \leq i \leq n.$$

Since $(x - \alpha_i)$ are distinct for all i, then the product $G(x)^{m-s}$ divides $D_{0,s}Q(x, R(x))$ too.

Conversely, suppose $G(x)^{m-s} \mid D_{0,s}Q(x, R(x))$ for all s with $0 \leq s < m$. Then each factor (shifted to the origin, i.e. $\alpha_i = 0$) $x^{m-s} \mid D_{0,s}c_i\varphi_i(x, R(x))$. This means that for some polynomial $U_s(x)$,

$$D_{0,s}Q(x, R(x)) = x^{m-s}U_s(x) \text{ with } 0 \leq s < m.$$

By Corollary 4.3.3, we can write

$$\begin{aligned} Q(x, y) &= \sum_s D_{0,s}Q(x, R(x))(y - R(x))^s \\ &= \sum_{s<m} D_{0,s}Q(x, R(x))(y - R(x))^s + \sum_{s\geq m} D_{0,s}Q(x, R(x))(y - R(x))^s \\ &= \sum_{s<m} x^{m-s}U_{(m-s)}(x)(y - R(x))^s + \sum_{s\geq m} D_{0,s}Q(x, R(x))(y - R(x))^s. \end{aligned}$$

Observe that a common factor $(y - R(x))^s$ has only terms of degree at least s since $x \mid R(x)$. Therefore, every term in $Q(x, y)$ has degree greater than m. □

This proves that the multiplicity and weighted degree conditions of the Guruswami-Sudan interpolation step are satisfied by $Q(x, y)$ if and only if there exists a polynomial $B_s(x) \in \mathbb{F}_q[x]$ such that

$$D_{0,s}Q(x, R(x)) = B_s(x)G(x)^{m-s} \text{ with } 0 \leq s < m. \tag{5.3.4}$$

and

$$\deg B_s(x) < l(n - k) - m\tau + s, \text{ with } 0 \leq s < m. \tag{5.3.5}$$

In the following, we give a reformulation of the Guruswami-Sudan interpolation conditions in terms of modules over a univariate polynomial ring, $\mathbb{F}_q[x]$.

Corollary 5.3.8 ([12])**.** Let $\beta = (\beta_1, \cdots, \beta_n) \in \mathbb{F}_q^n$ be a received word, l a designed list size, and m a multiplicity parameter. Then $Q(x, y) \in \mathbb{F}_q[x, y]$ is an interpolation polynomial if and only if it is of the form

$$Q(x, y) = \sum_{i=0}^{m-1} B_i(x)G(x)^{m-i}(y - R(x))^i + \sum_{i=m}^{l} B_i(x)(y - R(x))^i. \tag{5.3.6}$$

Proof. From Proposition 5.3.7, this follows. □

Note that Equation (5.3.6) can be expanded as

$$Q(x,y) = \sum_{a=0}^{l} \left[\sum_{i=0}^{m-1} B_i(x) G(x)^{m-i} (-R(x))^{i-a} \binom{i}{a} + \sum_{i=m}^{l} B_i(x) (-R(x))^{i-a} \binom{i}{a} \right] y^a \tag{5.3.7}$$

$$=: \sum_{a=0}^{l} Q_a(x) y^a \tag{5.3.8}$$

in which case we considered polynomials $B_i(x)$ as variables. This gives a nice parameterisation such that $Q(x,y)$ becomes a valid interpolation polynomial if and only if

$$\deg Q_a(x) < m(n-\tau) - a(k-1) \text{ for all } a \text{ with } 0 \le a \le l. \tag{5.3.9}$$

5.3.9 A Block-Hankel Matrix. We know that Equation (5.3.4) (with Equation (4.3.1)) can be recorded as

$$\sum_{t=s}^{l} \binom{t}{s} Q_t(x) (R(x))^{t-s} = B_s(x) G(x)^{m-s}, \text{ for all } s \text{ with } 0 \le s < m. \tag{5.3.10}$$

Let the reciprocal polynomials (see [11], [10]) be the following

$$\begin{aligned} \bar{R}(x) &= x^{n-1} R(x^{-1}), \\ \bar{G}(x) &= x^n G(x^{-1}) = \prod_{i=1}^{n} (1 - \delta_i x), \\ \bar{B}_s(x) &= x^{l(n-k)-m\tau-s-1} B(x^{-1}), \\ \Lambda_t(x) &= x^{N_t - 1} Q_t(x^{-1}) \end{aligned}$$

which, when inserted in Equation (5.3.10) yield

$$\sum_{t=s}^{l} \binom{t}{s} \Lambda_t(x) x^{(l-t)(n-k)} \bar{R}(x)^{t-s} = \bar{B}_s(x) \bar{G}(x)^{m-s}, \text{ for all } s \text{ with } 0 \le s < m. \tag{5.3.11}$$

Since $\bar{G}(0) \neq 0$, the following is well-defined

$$T^{(s,t)}(x) := \frac{\bar{R}(x)^{t-s}}{\bar{G}(x)^{m-s}}.$$

Thus, considering the $\binom{m+1}{2} n$ system of homogeneous linear equations, we obtain

$$\sum_{t=s}^{l} \binom{t}{s} \Lambda_t(x) x^{(l-t)(n-k)} T^{(s,t)}(x) \equiv \bar{B}_s(x) \bmod x^{m(n-\tau)+l(n-k)-s(n-1)}, \text{ for all } s \text{ with } 0 \le s < m. \tag{5.3.12}$$

Definition 5.3.10 (Guruswami-Sudan Syndrome [11])**.** The syndrome polynomials

$$S^{(0,0)}(x), S^{(0,1)}(x), \cdots, S^{(0,l)}(x), S^{(1,1)}(x), \cdots, S^{(m-1,l)}(x)$$

with

$$S^{(s,t)}(x) = \sum_{i=0}^{(m-s)n+N_t-1} S_i^{(s,t)} x^i$$

are given by

$$S_i^{(s,t)} = T_{i+(s+1+t(n-1)-mn)}^{(s,t)}, \text{ for } t = s, \cdots, l. \tag{5.3.13}$$

By Definition 5.3.10, the homogeneous system can be modified, considering weighted degree conditions, as

$$\sum_{t=s}^{l} \Lambda_t(x) S^{(s,t)}(x) \equiv \bar{B}_s(x) \bmod x^{m(n-\tau)+l(n-k)-s(n-1)}, \tag{5.3.14}$$

$$\text{where } \deg \bar{B}_s(x) < l(n-k) - m\tau + s, \text{ for all } s \text{ with } 0 \leq s < m. \tag{5.3.15}$$

When we consider the high degree terms in Equation (5.3.14), there are

$$\sum_{s=0}^{m-1} (m-s)n = \binom{m+1}{2}$$

homogeneous equations and they can be parameterised as

$$\sum_{t=s}^{l} \sum_{i=0}^{N_t-1} Q_t^i S_{j+i}^{(s,t)} = 0, 0 \leq j < (m-s)n, 0 \leq s < m, \tag{5.3.16}$$

where Q_t^i is a univariate polynomial (a coefficient).

Definition 5.3.11 (Hankel matrix [11])**.** An $m \times n$ matrix S is a Hankel matrix if $S_{i,j} = S_{i-1,j+1}$ $\forall i, j$ with $1 \leq i \leq m-1$, $0 \leq j < n-1$.

The linear system in Equation (5.3.16) gives a Block-Hankel matrix (each sub-matrix $S^{(s,t)}$ is a Hankel matrix).

5.3.12 Basis Transformation. As mentioned, solving the Key Equations is crucial in syndrome decoding. There is an efficient algorithm by P. Beelen and K. Brander [12] that solves the Key Equations of the Guruswami-Sudan interpolation problem. We will highlight how their technique guarantees improved complexity. To see how they solved the Key Equations, one should read Section 3 in [12].

From the Equation (5.3.7), Beelen and Brander [12] let the system

$$\begin{pmatrix} Q_0(x) \\ Q_1(x) \\ \vdots \\ Q_l(x) \end{pmatrix} = \mathcal{A} \begin{pmatrix} B_0(x) \\ B_1(x) \\ \vdots \\ B_l(x) \end{pmatrix} \tag{5.3.17}$$

where they defined $\mathcal{A}$ from Equation (5.3.7) as $\mathcal{A} := [\mathcal{A}_1 \mid \mathcal{A}_2]$ such that

$$[\mathcal{A}_1]_{a,i} = G(x)^{m-i}(-R(x))^{i-a}\binom{i}{a}, \text{ with } (a,i) \in \{0,1,\cdots,l\} \times \{0,1,\cdots,l\}$$
$$\text{and } [\mathcal{A}_2]_{a,i} = (-R(x))^{i-a}\binom{i}{a}, \text{ with } (a,i) \in \{0,1,\cdots,l\} \times \{m, m+1,\cdots,l\}.$$

By the weighted degree condition of $Q_a(x)$ (Equation (5.3.9)), the system (5.3.17) is equivalent to

$$\begin{pmatrix} Q_0(x) \\ x^{k-1}Q_1(x) \\ x^{2(k-1)}Q_2(x) \\ \vdots \\ x^{l(k-1)}Q_l(x) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & x^{k-1} & 0 & \cdots & 0 \\ 0 & 0 & x^{2(k-1)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & x^{l(k-1)} \end{pmatrix} \mathcal{A} \begin{pmatrix} B_0(x) \\ B_1(x) \\ B_2(x) \\ \vdots \\ B_l(x) \end{pmatrix}.$$

Define

$$\mathcal{B} := \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & x^{k-1} & 0 & \cdots & 0 \\ 0 & 0 & x^{2(k-1)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & x^{l(k-1)} \end{pmatrix} \mathcal{A}. \tag{5.3.18}$$

Basically, a solution to the system (5.3.17) is any vector

$$(Q_0(x), x^{k-1}Q_1(x), x^{2(k-1)}Q_2(x), \cdots, x^{l(k-1)}Q_l(x))^{\mathrm{T}}$$

in the $\mathbb{F}_q[x]$-span of the columns of $\mathcal{B}$ and have maximum degree less than $m(n-\tau)$.

Definition 5.3.13 ([12])**.** The maximum degree of a vector is the degree of its entry with the highest power, and the maximum degree of a collection of vectors is the sum of the maximum degrees of each vector.

For $0 \le i < m$, $i \le j \le l$, let $S^{(i,j)}(x)$ be the syndrome polynomial given by Definition 5.3.10. In other words, there is some polynomial $E^{(i,j)}(x)$ with

$$R^{j-i}(x) = E^{(i,j)}(x)G(x)^{m-i} + S^{(i,j)}(x)$$

such that $\deg S^{(i,j)}(x) < \deg G(x)^{m-i} = (m-i)n$. Also, define a matrix $\mathcal{U}$ such that for $(i,j) \in \{0,1,\cdots,l\} \times \{0,1,\cdots,l\}$

$$[\mathcal{U}]_{i,j} = \begin{cases} 1 & \text{if } i = j \text{ and } j < m, \\ 0 & \text{if } i \neq j \text{ and } j < m, \\ \binom{j}{i} E^{(i,j)}(x) & \text{if } i < m \text{ and } j \ge m, \\ \binom{j}{i} R(x)^{j-i} & \text{if } i \ge m \text{ and } j \ge m. \end{cases} \tag{5.3.19}$$

The matrix $\mathcal{U}$ is a good matrix for change of basis since it is upper triangular with determinant 1. Thus, the module spanned by the columns of $\mathcal{B}$ and $\mathcal{BU}$ are the same. Therefore, if a vector spans the columns of $\mathcal{B}$ also spans the columns of $\mathcal{BU}$ but with a lower maximum degree, we achieve a reduced complexity. Below we give a result from [12] that proves that a basis transformation resulting in an improved complexity is attained by their method of solving the Guruswami-Sudan interpolation step.

Proposition 5.3.14 ([12])**.** Let $\mathcal{B}$ be as in Equation (5.3.18), $\mathcal{U}$ be as in Equation (5.3.19). Then for $0 \leq a \leq l$ and $0 \leq i \leq l$,

$$[\mathcal{BU}]_{a,i} = \begin{cases} x^{a(k-1)} G(x)^{m-i} (-R(x))^{i-a} \binom{i}{a} & \text{if } a \leq i \text{ and } 0 \leq i < m, \\ -x^{a(k-1)} \sum_{h=a}^{m-1} (-R(x))^{h-a} \binom{h}{a} \binom{i}{h} S^{(h,i)}(x) & \text{if } a < i \text{ and } m \leq i \leq l, \\ x^{a(k-1)} & \text{if } a = i \text{ and } m \leq i \leq l, \\ 0 & \text{if } a > i. \end{cases} \tag{5.3.20}$$

Moreover, the maximum degree of $\mathcal{BU}$ is less than $(l+1)mn$.

Proof. We shall simplify using the fact that matrices $\mathcal{BU}$ and $\mathcal{B}$ has the same columns in their first m columns. We can write

$$\begin{aligned} \sum_{h=0}^{l} [\mathcal{A}]_{a,h} [\mathcal{U}]_{h,i} &= \sum_{h=0}^{m-1} [\mathcal{A}]_{a,h} [\mathcal{U}]_{h,i} + \sum_{h=m}^{l} [\mathcal{A}]_{a,h} [\mathcal{U}]_{h,i} \\ &= \sum_{h=0}^{m-1} G(x)^{m-h} (-R(x))^{h-a} \binom{h}{a} \binom{i}{h} E^{(h,i)}(x) + \sum_{h=m}^{l} (-R(x))^{h-a} \binom{h}{a} \binom{i}{h} R(x)^{i-h} \\ &= \sum_{h=0}^{m-1} (-R(x))^{h-a} \binom{h}{a} \binom{i}{h} (R(x)^{i-h} - S^{(h,i)}(x)) + R(x)^{i-a} \sum_{h=m}^{l} (-1)^{h-a} \binom{h}{a} \binom{i}{h} \\ &= R(x)^{i-a} \sum_{h=0}^{l} (-1)^{h-a} \binom{h}{a} \binom{i}{h} - \sum_{h=0}^{m-1} (-R(x))^{h-a} \binom{h}{a} \binom{i}{h} S^{(h,i)}(x). \end{aligned}$$

We know that

$$\sum_{h=0}^{l} (-1)^{h-a} \binom{h}{a} \binom{i}{h} = \begin{cases} 1 & \text{if } a = i, \\ 0 & \text{otherwise.} \end{cases}$$

Then

$$[\mathcal{AU}]_{a,i} = \begin{cases} G(x)^{m-i} (-R(x))^{i-a} \binom{i}{a} & \text{if } a \leq i \text{ and } 0 \leq i < m, \\ -\sum_{h=a}^{m-1} (-R(x))^{h-a} \binom{h}{a} \binom{i}{h} S^{(h,i)}(x) & \text{if } a < i \text{ and } m \leq i \leq l, \\ 1 & \text{if } a = i \text{ and } m \leq i \leq l, \\ 0 & \text{if } a > i. \end{cases}$$

Proving the last part, we got from the first part that for $m \leq i \leq l$ the maximum degree of $[\mathcal{BU}]_{a,i}$ is bounded:

$$\max_{0 \leq h \leq m-1}\{(h-a)(n-1)+(m-h)n\}+a(k-1) = mn - a(n-k) \leq mn.$$

By previous results, for $0 \leq i \leq m-1$ the maximum degree for column i in $\mathcal{BU}$ is less than $mn - i$. It follows that the maximum degree of $\mathcal{BU}$ is less than $(l+1)mn$. □

The method of writing the Guruswami-Sudan interpolation problem in terms of modules over a univariate polynomial ring resulted in a complexity gain. P. Beelen and K. Brander [12] reformulated and showed how to solve the problem, and presented an algorithm for improved complexity. Their algorithm can complete the interpolation step of the Guruswami-Sudan problem in $O(ml^4 n\log^2 n\log\log n)$ time. Note that the basis transformation results in a gain only if $l > m$ (it could be assumed tha $m/l \approx \kappa$ in Equation (5.1.2)).

6. Conclusion

The polynomial reconstruction technique ensures an improved complexity since Equation (5.2.7) is greater than Equation (5.1.1). We asserted in Section 3.2 that codes with large minimum distance are desired. This unequivocally points to the fact that a large decoding radius is preferred for better decoding. The gain in radius is at the expense of other complexities like the decoding time and the number of polynomials listed by the interpolation step. We saw in Theorem 5.2.3 that the list size and the time taken to enumerate the polynomials make no improvement to their counterparts in the Guruswami-Sudan algorithm.

In the case of basis transformation, the Guruswami-Sudan problem is reformulated in terms of modules over a univariate polynomial ring and the basis of the modules are replaced accordingly for improved complexity (decoding time) in the interpolation step. This is a nice technique in the sense that its gain is not at the expense of other complexities like the decoding radius and list size.

The polynomial reconstruction and the basis transformation could be coupled to decode Reed-Solomon codes. However, this will not ensure any newer improvement in the complexities. In fact, attaining a good reduction as required by Equation (5.2.6), the coupled system will complete the interpolation step in $O(ml^4n^{r+1}\log^2 n \log\log n)$ time (greater than $O(m^6n^3)$ time, the complexity for the interpolation step of the Guruswami-Sudan algorithm), where r is the number of reductions as in Theorem 5.2.2.

References

[1] Muhammad Taqi-ud-Din Al-Hilali and Muhammad Muhsin Khan. *Translation of the Meanings of the Noble Quran in the English Language*. 2018.

[2] Madhu Sudan. Decoding of reed solomon codes beyond the error-correction bound. *Journal of complexity*, 13(1):180–193, 1997.

[3] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Trans. Inform. Theory*, 45(6):1757–1767, 1999.

[4] VN Muralidhara and Sandeep Sen. Improvements on the johnson bound for reed–solomon codes. *Discrete Applied Mathematics*, 157(4):812–818, 2009.

[5] Lloyd R Welch and Elwyn R Berlekamp. Error correction for algebraic block codes, December 30 1986. US Patent 4,633,470.

[6] Sigal Ar, Richard J Lipton, Ronitt Rubinfeld, and Madhu Sudan. Reconstructing algebraic functions from mixed data. *SIAM Journal on Computing*, 28(2):487–510, 1998.

[7] Farzad Parvaresh and Alexander Vardy. Correcting errors beyond the guruswami-sudan radius in polynomial time. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 285–294. IEEE, 2005.

[8] Ralf Kotter. Fast generalized minimum-distance decoding of algebraic-geometry and reed-solomon codes. *IEEE Transactions on Information Theory*, 42(3):721–737, 1996.

[9] Ron M Roth and Gitit Ruckenstein. Efficient decoding of reed-solomon codes beyond half the minimum distance. *IEEE Transactions on Information Theory*, 46(1):246–257, 2000.

[10] Daniel Augot and Alexander Zeh. On the roth and ruckenstein equations for the guruswami-sudan algorithm. In *2008 IEEE International Symposium on Information Theory*, pages 2620–2624. IEEE, 2008.

[11] Alexander Zeh, Christian Gentner, and Daniel Augot. An interpolation procedure for list decoding reed–solomon codes based on generalized key equations. *IEEE Transactions on Information Theory*, 57(9):5946–5959, 2011.

[12] Peter Beelen and Kristian Brander. Key equations for list decoding of reed–solomon codes and how to solve them. *Journal of Symbolic Computation*, 45(7):773–786, 2010.

[13] Michael Alekhnovich. Linear diophantine equations over polynomials and soft decoding of reed-solomon codes. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 439–448. IEEE, 2002.

[14] Kwankyu Lee and Michael E O'Sullivan. List decoding of reed–solomon codes from a gröbner basis perspective. *Journal of Symbolic Computation*, 43(9):645–658, 2008.

[15] Jurgen Bierbrauer. *Introduction to Coding Theory*. Chapman and Hall/CRC, 2004.

[16] Raymond W Yeung. *Information Theory and Network Coding*. Springer Science & Business Media, 2008.

[17] Steven Roman. *Field Theory*, volume 158. Springer Science & Business Media, 2005.

[18] Robert J McEliece. The guruswami-sudan decoding algorithm for reed-solomon codes. *IPN progress report*, 42:153, 2003.

[19] Elwyn Berlekamp. *Algebraic Coding Theory*. World Scientific, 2015.

[20] Yuval Cassuto, Jehoshua Bruck, and Robert J McEliece. On the average complexity of reed–solomon list decoders. *IEEE Transactions on Information Theory*, 59(4):2336–2351, 2013.

[21] W Peterson. Encoding and error-correction procedures for the bose-chaudhuri codes. *IRE Transactions on Information Theory*, 6(4):459–470, 1960.

[22] Venkatesan Guruswami and Madhu Sudan. Reflections on improved decoding of reed-solomon and algebraic-geometric codes. *IEEE Information Theory Society Newsletter*, 52(1):6–12, 2002.

[23] Venkatesan Guruswami. List decoding of error-correcting codes, volume 3282 of. *Lecture Notes in Computer Science*, 2005.

[24] Ron Roth. *Introduction to Coding Theory*. Cambridge University Press, 2006.

[25] Thomas W Hungerford and Springer Algebra. Graduate texts in mathematics 73. *New York*, 1974.