ulm university universität
uulm

Algebraic Decoding over Finite and Complex Fields using Reliability Information

DISSERTATION

zur Erlangung des akademischen Grades eines

DOKTOR-INGENIEURS
(DR.-ING.)

der Fakultät für Ingenieurwissenschaften,
Informatik und Psychologie der Universität Ulm

von

Mostafa Hosni Mohamed

aus Riad - Saudi Arabien

Gutachter:	Prof. Dr.-Ing. Martin Bossert
	Prof. Dr.-Ing. Jürgen Freudenberger
Amtierender Dekan:	Prof. Dr. rer.nat. Frank Kargl

Ulm, 26.09.2017

To Nirvana and Hassan,
to whom I owe almost everything.

Preface

This dissertation presents recent results of my research activities at the Institute of Communications Engineering, Ulm University. Parts of the results have been published within the following articles in conference proceedings:

[1] M. H. Mohamed, Johan S. R. Nielsen, and Martin Bossert. Reduced List-Decoding of Reed–Solomon Codes Using Reliability Information. In *Proceedings of the 21st International Symposium on Mathematical Theory of Networks and Systems (MTNS)*, Groningen, the Netherlands, July 2014.

[2] M. H. Mohamed and Martin Bossert. A Chase-like Decoding Algorithm for Reed–Solomon Codes Based on the Extended Euclidean Algorithm. In *Proceedings of the 10th International ITG Conference on Systems, Communications and Coding (SCC)*, Hamburg, Germany, February 2015

[3] Henning Zörlein, Shrief Rizkalla, M. H. Mohamed, and Martin Bossert. Deterministic Compressed Sensing with Power Decoding for Complex Reed–Solomon Codes. In *Proceedings of the 10th International ITG Conference on Systems, Communications and Coding (SCC)*, Hamburg, Germany, February 2015.

[4] M. H. Mohamed, Henning Zörlein, and Martin Bossert. Recursive Enhancement of Intrinsic Soft Information for Complex Reed–Solomon Codes. In *Compressed Sensing Theory and its Applications to Radar, Sonar and Remote Sensing (CoSeRa)*, Aachen, Germany, September 2016.

[5] M. H. Mohamed, Sven Puchinger, and Martin Bossert. Guruswami–Sudan List Decoding for Complex Reed–Solomon Codes. In *Proceedings of the 11th International ITG Conference on Systems, Communications and Coding (SCC)*, Hamburg, Germany, February 2017.

Parts of my work have been supported by the German research council *Deutsche Forschungsgemeinschaft* (DFG) under Grant Bo 867/35-1.

Acknowledgements

This dissertation contains most of my work as PhD student at the Institute of Communications Engineering at Ulm University from June 2012 until July 2017. During these five years, I was honored to have the support of many people to whom I would like to express my gratitude.

First and foremost, I would like to thank my doctoral supervisor Martin Bossert for the continuous support and his trust in me to be part of his doctoral family. He provided me and all my colleagues with a great working environment full of motivation, creativity and excellence.

Furthermore, I would like to express my gratitude to Prof. Dr.-Ing. Jürgen Freudenberger from Hochschule Konstanz for taking the time to review my thesis. I would also like to thank Prof. Dr.-Ing. Christian Damm and Prof. Dr.-Ing. Stephan Reuter for being part of the colloquium.

I am also very thankful for the time I spent working with Antonia Wachter-Zeh, Vladimir Sidorenko, Johan S. H. Rosenkilde (né Nielsen) and Henning Zörlein for teaching me how to be a professional researcher through fruitful discussions and their always-available constructive feedback.

I would like to express my gratitude to all of my (former) colleagues, both doctoral candidates and staff members, who made my time in the institute worth while. Special thanks to my fantastic officemates: Susanne Sparrer for being so relentless in helping me become a better German speaker, and Sven Puchinger for restoring part of my faith in mathematics once more. I would like also to give thanks to those who helped me in proofreading parts of this thesis: Susanne Sparrer, Sven Müelich, Michael Schelling, Zaid Dhannoon, and Arthur Witt.

Last but not the least, my deepest gratitude and love belong to my lovely wife Menna for her support, superhuman patience, and occasional well-timed advice. I love you so much.

A shout-out to my little Maya. I love you, my princess.

Mostafa H. Mohamed
Ulm, September 2017

Abstract

In this dissertation, new algebraic decoding algorithms for Reed–Solomon codes are developed, all of which use reliability information. The two main scenarios are Reed–Solomon codes defined over finite fields and the complex field. For each scenario, we introduce two new algorithms: a syndrome-based and an interpolation-based decoder.

For the first scenario, a syndrome-based method which depends on an intermediate decoding result obtained by the extended Euclidean algorithm is investigated. This method is suitable only for high-rate codes, where one or two additionally correctable errors are valuable. The second method in this scenario, utilizes the same intermediate result to perform an interpolation step similar to the Wu algorithm but with a reduced number of interpolation points. As a results, the complexity of the interpolation step is reduced considerably.

The second scenario is the decoding of complex-valued Reed–Solomon codes, which recently gained attention in deterministic Compressed Sensing schemes. They allow the use of known algebraic decoding algorithms for sparse vector reconstruction. It is also possible to extract and exploit intrinsic reliability information. The first decoding method for this scenario performs a multi trial error/erasure decoding procedure to enhance the quality of the reliability information. While the other is a list decoder based on both the Guruswami–Sudan algorithm and generelized minimum distance decoding.

The performance of all the aforementioned algorithms has been investigated and compared with similar state-of-the-art algorithms. Without exceptions, their performance surpasses that of their counterparts.

The second part of this work has been supported by the German research council *Deutsche Forschungsgemeinschaft* (DFG) under Grant Bo 867/35-1.

Contents

List of Acronyms and Symbols

Acronyms

ADSL	Asymmetric Digital Subscriber Line
AWGN	Additive White Gaussian Noise
BCASC	Best Complex Antipodal Spherical Code
BCH	Bose–Chaudhuri–Hocquenghem
BMA	Berlekamp–Massey Algorithm
BPSK	Binary Phase shift Keying
CAD	Continuity Assisted Decoding
CoSIP	Compressed Sensing in Information Processing
CRS	Complex Reed–Solomon
CS	Compressed Sensing
DFG	Deutsche Forschungsgemeinschaft
DFT	Discrete Fourier Transform
EEA	Extended Euclidean Algorithm
GMD	Generalized Minimum Distance
GZ	Gorenstein–Zierler
GS	Guruswami–Sudan
IDFT	Inverse Discrete Fourier Transform
IRS	Interleaved Reed–Solomon
KV	Kötter–Vardy
LFSR	Linear Feedback Shift Register
MDS	Maximum Distance Separable
mRR	modified Roth–Ruckenstein
OFDM	Orthogonal Frequency-Division Multiplexing
OMP	Orthogonal Matching Pursuit
PD	Power Decoding
REA	Recursive Enhancement Algorithm
RLD	Reduced-List Decoder
RR	Roth–Ruckenstein
RS	Reed–Solomon
SNR	Signal-to-Noise Ratio

Symbols

$\mathbb{C}$	Field of complex numbers
$\mathbb{F}$	Finite field
$\mathbb{F}_p$	Prime field
$\mathbb{F}_{p^m}$	Extension field
α	Primitive element
$\mathbb{F}_p[x]$	Set of all polynomials over $\mathbb{F}_p$
${}^{\mathbb{F}_q[x]}/_{(x^n-1)}$	Ring of polynomials over $\mathbb{F}_p$ and degree $< n$
$\mathcal{F}[\cdot]$	DFT
$\mathcal{F}^{-1}[\cdot]$	IDFT
$\mathrm{wt_H}$	Hamming weight
d_H	Hamming distance
d_E	Euclidean distance
d_G	Generalized distance
$\boldsymbol{a} \in {\mathbb{F}_p}^n$	Vector over $\mathbb{F}_p$ of length n
$\boldsymbol{a}(x) \in {}^{\mathbb{F}_q[x]}/_{(x^n-1)}$	Polynomial over $\mathbb{F}_p$ and degree $< n$
$\boldsymbol{A} \in \mathbb{F}_p^{k \times n}$	Matrix over $\mathbb{F}_p$ with k rows and n columns
$\mathcal{C} \subset \mathbb{F}_p^n$	A code
$\boldsymbol{c} \in \mathcal{C}$	A codeword
n	Code length
k	Code dimension
R	Code rate
$\mathcal{C}^\perp \subset \mathbb{F}_p^n$	Dual of $\mathcal{C}$
$\mathcal{RS}$	Reed–Solomon code
$\mathcal{IRS}$	Interleaved Reed–Solomon code
$\boldsymbol{G}$	Generator matrix
$\boldsymbol{H}$	Parity check matrix
$\boldsymbol{g}(x)$	Generator polynomial
$\boldsymbol{h}(x)$	Parity check polynomial
$\boldsymbol{r}$	Received vector
$\boldsymbol{e}$	Error vector
Ψ	Set of error locations
$t = \lvert\Psi\rvert$	Number of errors
$\mathcal{E}(n, t)$	Error code
$\boldsymbol{\Gamma}(x)$	Error generator polynomial
$\boldsymbol{\Lambda}(x)$	Error locator (parity check) polynomial
$\boldsymbol{\Omega}(x)$	Error evaluator polynomial

$\boldsymbol{S}$ ..... Syndrome
$\tau = \lfloor {}^{d-1}/_{2} \rfloor$ ..... Classical decoding radius
τ_p ..... Power decoding radius
τ_{GS} ..... Guruswami–Sudan decoding radius
τ_{Wu} ..... Wu decoding radius
$\boldsymbol{Q}(x, y)$ ..... Bivariate interpolation polynomial
$\boldsymbol{Q}(x, y, z)$ ..... Trivariate interpolation polynomial
$\deg_{(w_x, w_y, w_z)} \boldsymbol{Q}(x, y, z)$ ..... The (w_x, w_y, w_z)-weighted degree of $\boldsymbol{Q}(x, y, z)$
$\boldsymbol{\eta}$ ..... Reliability vector of $\boldsymbol{r}$
$\boldsymbol{\zeta}$ ..... Noise vector of $\boldsymbol{r}$
$\boldsymbol{\zeta}_S$ ..... Noise vector of syndrome $\boldsymbol{S}$
$[\mu/\nu]_{\boldsymbol{f}}(x)$ ..... Padé approximation of a function $\boldsymbol{f}(x)$

1

Introduction and Motivation

RELIABLE transmission of information has always been a critical requirement in any communication system. There is no doubt, that if a communication system is not reliable enough for its users' needs, it will become extinct in due time. Whether the signal is some written text, a flash of light, a coloured banner or even a smoke signal, the receiver should be able to recover the original signal without errors. Entering the digital world has changed many things. Bits became our representation of information. To save power wasted on retransmissions, error correction at the receiver became more imperative. As a result, coding theory was born. After that, Shannon [Sha48] proved that there exists a way of having an error-free transmission through coding.

There exists a plethora of different codes used for the purpose of error correction, both linear and non-linear. Linear codes are more favoured due to the existence of efficient decoding algorithms. Popular linear codes used in error correction include the Hamming [Ham50], Bose–Chaudhuri–Hocquenghem (BCH) [BRC60] and Reed–Solomon (RS) [RS60] codes and many more. More than 50 years after they were invented, RS codes are still one of the most popular codes used in error correction. They have gained the attention of many researchers due to their elegant structure based on the Discrete Fourier Transform (DFT). They even found their way to a lot of day-to-day life applications such as [WB94]: Storage devices like compact disks and spread-spectrum systems like frequency hopping. They are also used in deep space telecommunication systems like the ones on the *Voyager* interstellar missions. When defined over complex fields, it was recently discovered that RS codes can be used in the application of deterministic Compressed Sensing (CS). This discovery led to the possibility of funding some of the work presented in this dissertation by the German research council *Deutsche Forschungsgemeinschaft* (DFG) under the priority program '*SPP 1798: CS in Information Processing*' (CoSIP) with

the project title '*Complex-valued Reed-Solomon Codes for Deterministic CS*'. In this scenario, where codeword symbols are complex-value, numerical inaccuracies such as quantization errors and the finite precision of computations start to affect the decoding process and the robustness of a decoder against such inaccuracies should be observed.

There are many methods which can be used in order to decode RS codes. The optimal decoding method is maximum likelihood decoding [MS88]. Using this method, the probability of making a decoding mistake is minimized; thus it provides the best possible performance in comparison to other methods. Unfortunately, this method is computationally expensive since its complexity increases exponentially with the size of the code. A more practical approach would be to use suboptimal decoding methods like the minimum distance decoding. They provide adequate error correction performance with polynomial-time complexity. Algorithms using the minimum distance decoding method can be distinguished into two groups: First, syndrome-based decoders such as the Extended Euclidean Algorithm (EEA) [MS88, p. 362], the Berlekamp–Massey Algorithm (BMA) [Ber68, Chapter 7] and the Peterson algorithm [PW72]. Secondly, interpolation-based decoders such as the Welch-Berlekamp [WB86], the Sudan [Sud97], the Guruswami–Sudan (GS) [GS99] and the Wu [Wu08] algorithms.

With the help of reliability information, we can enhance the performance of these algorithms with a not-so-large computational cost. There are many forms of reliability information, it can be either given by an outside source 'genie' or can be calculated at the receiver. The way the reliability information is utilized differs from one algorithm to another. For instance, when dealing with binary codes, the Chase [Cha72] algorithm flips the positions with lowest reliability, while the Dorsch [Dor74] algorithm uses positions with highest reliability to perform re-encoding. For non-binary codes, Generalized Minimum Distance (GMD) decoding [For66] introduces erasures at positions with low reliability to remove their influence on the decoding process. There is also the Kötter–Vardy (KV) algorithm [KV03], an extension to the GS algorithm, where the reliability determines the multiplicity of the points used in the interpolation and their influence on the decoding result.

Using some of these algorithms combined with the presence of reliability information and error/erasure decoding, the goal of this dissertation is to show the possibility of providing new decoders, which achieve superior performance levels for a reasonable computational complexity. Specifically, we present new algorithms for decoding RS codes in two scenarios, where the codes are defined over finite fields and the complex field. For each scenario, we present two algo-

rithms: A syndrome-based algorithm and another which is interpolation-based. To measure the performance of the introduced algorithms, we perform numerical simulations and compare their performance with those of already established algorithms.

The structure of the dissertation is as follows: In **Chapter 2**, we provide the basic aspects and definitions of fields and the different metrics to be used in the following chapters. At the end of this chapter, the definition of RS codes is given.

The various classical RS decoding algorithms upon which the new algorithms are built, are explained in **Chapter 3**. We also explain error/erasure decoding, the definition of an erasure and GMD decoding.

In **Chapter 4**, we discuss the decoding of RS codes over the finite fields using reliability information. We first introduce two approaches for the calculation of the reliability information, which is then used in two decoders. The first decoder is the Chase-like decoder [MB15], which gets its name and idea from the Chase algorithm in [Cha72]. The second decoder is the Reduced-List Decoder (RLD) [**?**], which is based on the Wu algorithm [Wu08]. The performance of both decoders is evaluated through numerical simulations and compared to the performance of algorithms like GMD and KV.

In **Chapter 5**, we discuss the decoding of RS codes over the complex fields using reliability information. We first explain the relation between coding theory and CS, since the decoders in this scenario are used for sparse vector reconstruction. We also present new insights on the calculation of built-in reliability information and its properties. Again, we present two different decoders which utilize the calculated reliability information. The first decoder is the Recursive Enhancement Algorithm (REA) [MZB16], which exploits the behaviour of the calculated reliability information in order to enhance it using a GMD-like procedure. The second decoder is a GS-based GMD decoder [MPB17], which is the result of the successful attempt to make the GS algorithm usable for decoding complex-valued RS codes. The performance of both decoders is evaluated through numerical simulations and compared to the performance of a CRS decoding algorithm established in [Zö15] as well as the popular CS Orthogonal Matching Pursuit (OMP) algorithm [CBL89].

Finally, we summerize and conclude these contributions in **Chapter 6**.

2

Basic Definitions

ERROR correction is today an essential part of any communication system to achieve reliable transmission. An overhead added to a block of information before transmission provides the receiver with a chance to correct corrupted data, making systems more efficient. The process of adding redundancy to your information is called encoding. While recovering the information at the receiver is called decoding. Block codes are codes where uncoded messages as well as their encoded counterparts (codewords) have a predefined size. If any linear combination of these codewords results in another codeword from the same code, then they are called linear block codes. Famous linear block codes include Hamming [Ham50], Bose–Chaudhuri–Hocquenghem (BCH) [BRC60] and Reed–Solomon (RS) [RS60] codes. In this dissertation, the attention is focused on the latter.

This chapter covers the basic aspects and definitions of linear block codes in general and those of RS codes in particular. Also metrics that are used in the decoding procedures in the following chapters are introduced.

2.1 Finite Fields

Finite fields are also named Galois fields after the French mathematician Évariste Galois, who had many contributions to algebra. A finite field $\mathbb{F}$ is a set of integers on which the operations addition and multiplication are defined, while satisfying the axioms of a field [LN02].

Definition 2.1 (Prime field)
For a prime p, the prime field $\mathbb{F}_p$ is the set $\{0, 1, \ldots, p-1\}$ of integers. Let α be the primitive element of $\mathbb{F}_p$, such that all the non-zero elements of the field can be obtained from powers of α.

A polynomial over $\mathbb{F}_p$ can be expressed in the form $\boldsymbol{a}(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}$, where the degree $n-1$ is a non-negative integer and the coefficients $a_0, a_1, \ldots, a_{n-1} \in \mathbb{F}_p$. The set of polynomials over $\mathbb{F}_p$ is denoted as $\mathbb{F}_p[x]$. An irreducible polynomial is a polynomial having no roots in $\mathbb{F}_p$ and can not be factored to polynomials of a smaller degree.

Definition 2.2 (Extension field)
Let $q = p^m$, where p is a prime. An extension field $\mathbb{F}_q$ is a field containing q elements and $\mathbb{F}_p$ is considered as its base field. The primitive element α generating the field is a root of any primitive polynomial $\boldsymbol{p}(x) \in \mathbb{F}_p[x]$ with degree m.

In this work, polynomials are sometimes expressed as vectors and vice versa. This bijective relation is as follows:

$$\boldsymbol{a} = (a_0, a_1, \ldots, a_{n-1}) \quad \longleftrightarrow \quad \boldsymbol{a}(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}. \tag{2.1}$$

The set of all possible vectors of length n with coefficients from the field $\mathbb{F}_q$ can be denoted as $\mathbb{F}_q^n$. While a polynomial ring of all polynomials having a degree smaller than n can be denoted as ${}^{\mathbb{F}_q[x]}\!/_{(x^n-1)}$.

Definition 2.3 (Discrete Fourier Transform (DFT) [Bos99])
For a given polynomial $\boldsymbol{A}(x) \in {}^{\mathbb{F}_q[x]}\!/_{(x^n-1)}$, the DFT $\mathcal{F}[\boldsymbol{A}(x)] = \boldsymbol{a}(x) = a_0 + \cdots + a_{n-1}x^{n-1}$ is defined as:

$$a_j = \boldsymbol{A}(\alpha^j), \quad j = 0, \ldots, n-1, \tag{2.2}$$

and the Inverse DFT (IDFT) $\mathcal{F}^{-1}[\boldsymbol{a}(x)] = \boldsymbol{A}(x)$ as:

$$A_j = n^{-1}\boldsymbol{a}(\alpha^{-j}), \quad j = 0, \ldots, n-1. \tag{2.3}$$

2.2 Metrics

For a code $\mathcal{C}$ (defined in Section 2.3), the measure of performance is usually measured by the number of errors it is able to correct. This has direct relation to the minimum distance between its codewords. There exist many metrics to measure this distance including but not limited to Hamming metric, rank metric, Euclidean metric and many more. This metric is chosen depending on the application at hand.

For a distance function $d(\boldsymbol{a}, \boldsymbol{b})$ to be considered as a metric, the following properties must be satisfied for any 3 vectors $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathbb{F}_q^n$ [Rot06]:

- $d(\boldsymbol{a}, \boldsymbol{b}) \geq 0$,
- $d(\boldsymbol{a}, \boldsymbol{b}) = 0$, iff $\boldsymbol{a} = \boldsymbol{b}$,
- $d(\boldsymbol{a}, \boldsymbol{b}) = d(\boldsymbol{b}, \boldsymbol{a})$,
- $d(\boldsymbol{a}, \boldsymbol{b}) \leq d(\boldsymbol{a}, \boldsymbol{c}) + d(\boldsymbol{b}, \boldsymbol{c})$.

In the following chapters, only two metrics are used, namely the Hamming and the Euclidean metrics. The former is used in Chapter 4 and the latter in Chapter 5.

2.2.1 Hamming Distance

The Hamming distance was introduced by Richard Hamming in [Ham50], where he also introduced Hamming codes. This distance measuring is one of the most used in digital communications and coding theory. It is usually used when the vectors in question are defined over finite fields. Before defining the Hamming distance, first we introduce the Hamming weight.

Definition 2.4 (Hamming weight)
For a given vector $\boldsymbol{a} \in \mathbb{F}_q^n$, the Hamming weight is defined as the number of non-zero elements in $\boldsymbol{a}$

$$\mathrm{wt_H}(\boldsymbol{a}) = \#\{j : a_j \neq 0\}, \tag{2.4}$$

where $\#\{\cdot\}$ denotes the cardinality of a set.

Definition 2.5 (Hamming distance)
For two given vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}_q^n$, the Hamming distance is defined as the number of positions where $\boldsymbol{a}$ and $\boldsymbol{b}$ are different

$$d_{\mathrm{H}}(\boldsymbol{a}, \boldsymbol{b}) = \mathrm{wt_H}(\boldsymbol{a} - \boldsymbol{b}) = \#\{j : a_j \neq b_j\}. \tag{2.5}$$

2.2.2 Euclidean Distance

Named after the famous Greek mathematician Euclid, the Euclidean metric is used to measure the distance between two points in an n-dimensional space [Bos99]. This distance is also called the Euclidean norm. This metric is used in applications where vectors over the complex field $\mathbb{C}$ are used, such as Compressed Sensing (CS) [EK12].

Definition 2.6 (Euclidean distance)
For two given vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{C}^n$, the Euclidean distance is defined as:

$$d_{\mathrm{E}}(\boldsymbol{a}, \boldsymbol{b}) = \sqrt{\sum_{j=0}^{n-1} (a_j - b_j)^2}. \tag{2.6}$$

There exists also the squared Euclidean distance where the square root is simply omitted. However, it can not be considered a metric as it does not satisfy all the properties of a metric.

2.3 Linear Block Codes

Coding is used to achieve, with high probability, an error-free communication between two points (sender and receiver) over a noisy channel. This noisy channel introduces errors to the transmitted data resulting in corrupted data at the receiver side. Coding is used to correct these errors introduced by the channel. The method of decoding and error correction depends on the type of code used in the system. This dissertation focuses on the use of two variants of RS codes, which fall under the category of cyclic linear block codes. Block codes indicate the encoding process is simply a mapping from blocks of information with length k to blocks of codewords with length n. The ratio between the information length and codeword length is called the rate of the code $R = \frac{k}{n}$. Let the information block be $\boldsymbol{u} = (u_0, u_1, \ldots, u_{k-1}) \in \mathbb{F}_q^k$.

Definition 2.7 (Linear block code [MS88])
A code $\mathcal{C} \subset \mathbb{F}_q^n$ is considered linear if for any two codewords $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{C}$ the following is satisfied:

$$a\boldsymbol{c}_1 + b\boldsymbol{c}_2 \in \mathcal{C}, \qquad a, b \in \mathbb{F}_q.$$

A linear code is a subspace of $\mathbb{F}_q^n$ of size $\#\mathcal{C} = q^k$. The linear mapping from information to codewords is defined by a basis of the subspace. This basis is referred to as the *generator matrix* $\boldsymbol{G}$ such that:

$$\boldsymbol{c} = \boldsymbol{u}\boldsymbol{G}, \quad \boldsymbol{G} \in \mathbb{F}_q^{k \times n}. \tag{2.7}$$

Definition 2.8 (Dual code [MS88])
The dual code $\mathcal{C}^\perp$ is the orthogonal complement of $\mathcal{C}$ and is defined such that:

$$\mathcal{C}^\perp = \{\boldsymbol{s} : \forall \boldsymbol{c} \in \mathcal{C}, \langle \boldsymbol{s}, \boldsymbol{c} \rangle = 0\}, \quad \mathcal{C}^\perp \subset \mathbb{F}_q^n, \tag{2.8}$$

where $\langle \cdot, \cdot \rangle$ denotes a scalar product between two vectors.

Having the definition of the dual code, we can introduce the *parity check matrix* $\boldsymbol{H}$, which is a basis of the dual code $\mathcal{C}^{\perp}$. $\boldsymbol{H}$ is an $(n-k) \times n$ matrix. As the name gives it away, this matrix is used to check whether a vector is a codeword or not since

$$\forall \boldsymbol{c} \in \mathcal{C}, \quad \boldsymbol{c}\boldsymbol{H}^T = \boldsymbol{0}. \tag{2.9}$$

It is also worth noting that the matrix multiplication of both matrices $\boldsymbol{G}$ and $\boldsymbol{H}^T$ yields a zero matrix.

$$\boldsymbol{G}\boldsymbol{H}^T = \boldsymbol{0}. \tag{2.10}$$

For a linear code, the minimum Hamming distance of the code is equal to the minimum Hamming weight of all codewords.

$$\begin{aligned} d(\mathcal{C}) \coloneqq d_{\mathrm{H}} &= \min_{\substack{\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{C} \\ \boldsymbol{c}_1 \neq \boldsymbol{c}_2}} d_{\mathrm{H}}(\boldsymbol{c}_1, \boldsymbol{c}_2), \\ &= \min_{\substack{\boldsymbol{c} \in \mathcal{C} \\ \boldsymbol{c} \neq \boldsymbol{0}}} \mathrm{wt}_{\mathrm{H}}(\boldsymbol{c}). \end{aligned} \tag{2.11}$$

The distance d_{H} is bounded by many famous bounds such as the Hamming bound and the Singleton bound. For the rest of the thesis, the Hamming distance is denoted either as d_{H} or d for simplicity.

Theorem 2.1 (Hamming bound [Bos99])
For a code $\mathcal{C}$ with length n and dimension k over the Field $\mathbb{F}_q$, the distance d is bounded by the following inequality:

$$q^k \left(1 + (q-1)\binom{n}{1} + (q-1)^2\binom{n}{2} + \cdots + (q-1)^{\tau}\binom{n}{\tau}\right) \leq q^n, \tag{2.12}$$

where $\tau = \lfloor {}^{d-1}/_{2} \rfloor$.

The Hamming bound shows the maximum radius non overlapping spheres can have if drawn around each codeword such that the total number of vectors in all the spheres does not exceed the size of $\mathbb{F}_q^n$. Satisfying this bound with equality means that all elements of $\mathbb{F}_q^n$ are contained in these spheres, thus decoding is always possible since any vector in $\mathbb{F}_q^n$ is at a distance less than $d/2$ from a codeword. In that case, this code is called a *perfect code*.

Theorem 2.2 (Singleton bound [MS88])
For a code $\mathcal{C}$ with length n and dimension k, the distance d is bounded by the following inequality:

$$d \leq n - k + 1.$$

The singleton bound relates the size of the redundancy $(n-k)$ to the distance. The distance can never be larger than the number of extra symbols that were added in the encoding procedure. Satisfying this bound with equality indicates that the code makes perfect use of the redundancy provided. This code is then called Maximum Distance Separable (MDS) [MS88], where any k positions of a vector determine the rest of a codeword. One famous example of codes being MDS are RS codes.

2.4 Reed–Solomon Codes

Reed–Solomon codes were first introduced in [RS60]. Since then, they gained a huge popularity as error correcting codes. More than 50 years after they were defined, they are still used on a daily basis in many communication systems, storage devices and other various scientific purposes as well as consumer applications [WB94]. There are many ways to define RS codes. Either using the original definition introduced in [RS60] or the generalized form as in [Del75]. They are also mentioned in many text books with different points of view. For example, in [Bos99] they are defined using polynomials, while in [Rot06] they are defined using vectors and matrices. In this dissertation, without loss of generality, the classical definition based on polynomials is used.

Definition 2.9 (Reed–Solomon codes [Bos99])
Let n and k be positive integers fulfilling $k < n$. The code $\mathcal{RS}(n,k)$ of length n and dimension k is the set

$$\mathcal{RS} = \left\{ \left(\boldsymbol{C}\left(\alpha^0\right), \ldots, \boldsymbol{C}\left(\alpha^{n-1}\right)\right) \middle| \, \boldsymbol{C}(x) \in \mathbb{F}_q[x],\, \deg \boldsymbol{C}(x) < k \right\}, \tag{2.13}$$

where α is a primitive element of $\mathbb{F}_q$ and the minimum distance of the code is $d = n - k + 1$.

Therefore, the relation between the codeword $\boldsymbol{c} = (c_o, c_1, \ldots, c_{n-1})$ and the information $\boldsymbol{C} = (C_0, C_1, \ldots C_{k-1}, 0, \ldots, 0)$ can be considered as a DFT relation(see Definition 2.3) such that $\boldsymbol{c}(x) = \mathcal{F}[\boldsymbol{C}(x)]$.

As a result, the generator matrix $\boldsymbol{G}_{\mathrm{RS}}$ has the following form

$$\boldsymbol{G}_{\mathrm{RS}} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha^1 & \alpha^2 & \cdots & \alpha^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^{k-1} & \alpha^{2(k-1)} & \cdots & \alpha^{(n-1)(k-1)} \end{pmatrix}, \tag{2.14}$$

and the parity check matrix $\boldsymbol{H}_{\mathrm{RS}}$

$$\boldsymbol{H}_{\mathrm{RS}} = \begin{pmatrix} 1 & \alpha^k & \alpha^{2k} & \cdots & \alpha^{(n-1)k} \\ 1 & \alpha^{k+1} & \alpha^{2(k+1)} & \cdots & \alpha^{(n-1)(k+1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^{n-1} & \alpha^{2(n-1)} & \cdots & \alpha^{(n-1)(n-1)} \end{pmatrix}. \tag{2.15}$$

Note that both matrices are partial DFT matrices in a Vandermonde matrix form. In the matrix form, the vector $\boldsymbol{C}$ is considered the information vector such that $\boldsymbol{c} = \boldsymbol{C} \cdot \boldsymbol{G}_{\mathrm{RS}}$. Another way of encoding in the polynomial form is by multiplying the information by the *generator polynomial* such that $\boldsymbol{c}(x) = \boldsymbol{u}(x)\boldsymbol{g}(x)$ and $\boldsymbol{g}(x)$ is defined as

$$\boldsymbol{g}(x) = \prod_{j=n-k}^{n-1} (x - \alpha^{-j}), \tag{2.16}$$

and the *parity check polynomial* $\boldsymbol{h}(x)$ as:

$$\boldsymbol{h}(x) = \prod_{j=0}^{k-1} (x - \alpha^{-j}), \tag{2.17}$$

with the relation between both polynomials:

$$\boldsymbol{g}(x)\boldsymbol{h}(x) = x^n - 1. \tag{2.18}$$

Interleaved Reed–Solomon Codes

Interleaved RS (IRS) codes are linear block codes which are constructed through the interleaving of codewords from one or more RS codes. IRS codes are used in many applications where burst errors are more common such as Asymmetric Digital Subscriber Line (ADSL) [ITU99] and Compact Disks (CDs) [WB94]. The definition of IRS codes is as follows

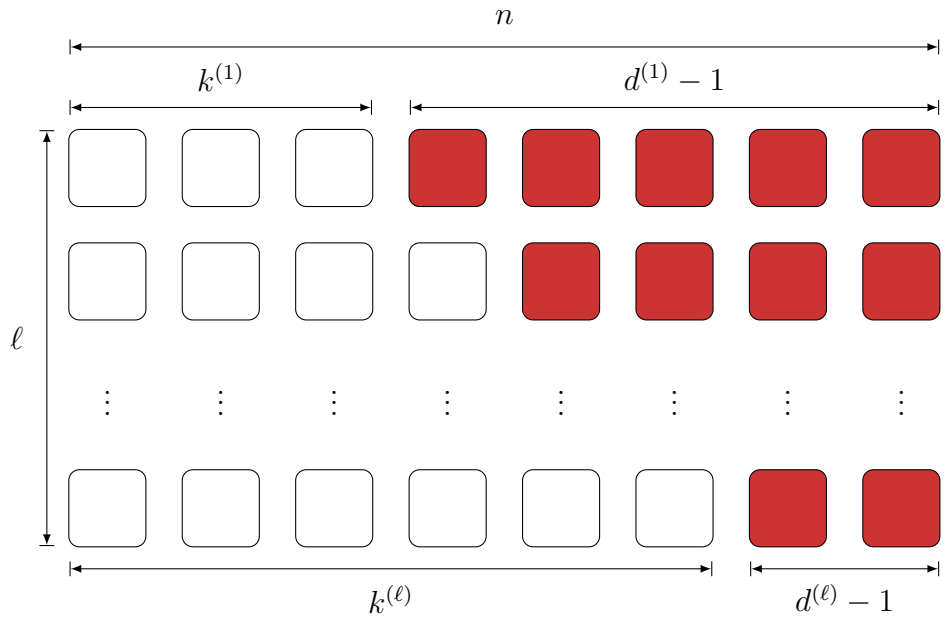

Figure 2.1: Heterogeneous IRS codes, where each row represents a codeword. The white and red boxes represent information and redundancy symbols respectively.

Definition 2.10 (Interleaved Reed–Solomon codes)
Let $\mathcal{RS}^{(h)} = \mathcal{RS}(n, k^{(h)})$ for $h = 1, \ldots, \ell$ be a group of ℓ RS codes of length n with each having a dimension of $k^{(h)}$. An IRS code can be defined as the following set of matrices

$$\mathcal{IRS} \coloneqq \left\{ \begin{pmatrix} \boldsymbol{c}^{(1)} \\ \boldsymbol{c}^{(2)} \\ \vdots \\ \boldsymbol{c}^{(\ell)} \end{pmatrix} : \boldsymbol{c}^{(h)} \in \mathcal{RS}^{(h)},\ h = 1, \ldots, \ell \right\}. \tag{2.19}$$

An IRS codes is called homogeneous if the dimensions $k^{(h)}$ $h = 1, \ldots, \ell$ are all equal, otherwise, it is called heterogeneous. Figure 2.1 shows an example of a heterogeneous IRS code, where each row represents a codeword. The white and red boxes represent information and redundancy symbols respectively. Such structure offers the possibility of collaborative decoding at the receiver [SSB06]. thus, allowing error correction beyond half the minimum distance.

Although, we do not deal with IRS codes directly, the idea of collaborative decoding of multiple RS codewords is used in one of the decoding methods presented in Chapter 3.

3

Decoding Reed–Solomon Codes

BASED on Definition 2.9 of RS codes, we proceed in explaining different methods of decoding these codes. Let $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$ be a received word where $\boldsymbol{c} \in \mathcal{RS}(n, k)$ and $\boldsymbol{e} \in \mathbb{F}_q^n$ is an error vector. The positions with non-zero elements are denoted by the set $\Psi = \{i : e_i \neq \mathbf{0}\}$ and the number of errors $\#\Psi = t$. The goal of decoding is to recover the codeword $\boldsymbol{c}$ from the received vector $\boldsymbol{r}$. Different decoders use different algebraic methods and techniques to achieve this goal. Some use the syndrome $\boldsymbol{S}$ (explained in Section 3.1), while others use the received word $\boldsymbol{r}$ directly. The maximum number of errors that the decoder can correct is called decoding radius τ. Classical decoders can decode up to half the minimum distance of the code

$$\tau = \lfloor (d-1)/2 \rfloor. \tag{3.1}$$

If t is larger than this radius, these decoders either give a wrong output or fail to give any at all. While other more complex ones go beyond this conventional radius, this comes at a cost. They either no longer produce a unique output or retain an increased probability of failure. This chapter concentrates on the different variants of RS decoders utilized in later chapters.

3.1 Syndrome-based Decoding

The *syndrome* is defined as follows:

$$\boldsymbol{S} = \boldsymbol{r}\boldsymbol{H}^T = (\boldsymbol{c} + \boldsymbol{e})\boldsymbol{H}^T = \boldsymbol{e}\boldsymbol{H}^T, \tag{3.2}$$

where $\boldsymbol{H} = \boldsymbol{H}_{\mathrm{RS}}$. From Equation (3.2), it is clear that the syndrome is considered to be the closest obtainable relative to the unknown error vector $\boldsymbol{e}$. It can

also be obtained in its polynomial form by utilizing the IDFT (see Definition 2.3).

$$\boldsymbol{R}(x) = \mathcal{F}^{-1}[\boldsymbol{r}(x)] = \mathcal{F}^{-1}[\boldsymbol{c}(x) + \boldsymbol{e}(x)] = \boldsymbol{C}(x) + \boldsymbol{E}(x). \tag{3.3}$$

Since $C_k, \ldots, C_{n-1}$ are equal to 0, the following applies:

$$\begin{aligned} \boldsymbol{S} = (S_0, \ldots, S_{n-k-1}) \quad &\longleftrightarrow \quad \boldsymbol{S}(x) = S_0 + \cdots + S_{n-k-1}x^{n-k-1} \\ S_0 &= R_k = E_k, \\ S_1 &= R_{k+1} = E_{k+1}, \\ &\vdots \\ S_{n-k-1} &= R_{n-1} = E_{n-1}. \end{aligned} \tag{3.4}$$

Syndrome-based decoders divide the decoding process into two steps: First, finding the error locations, then calculating the error value. The location of non-zero elements in $\boldsymbol{e}$ is determined by the *error locator polynomial* $\boldsymbol{\Lambda}(x)$, which can be also regarded as the parity check polynomial of an *error code* to which an error codeword $\boldsymbol{E}(x)$ belongs.

Definition 3.1 (Error Code $\mathcal{E}(n,t)$)
Given the error polynomial $\boldsymbol{e}(x) = \sum_{i \in \Psi} e_i x^i$*, then the* (n, τ) *cyclic code over* $\mathbb{F}_q$ *with generator polynomial*

$$\boldsymbol{\Gamma}(x) = \prod_{i \in [0,n-1] \setminus \Psi} (x - \alpha^i) \tag{3.5}$$

and the parity check polynomial

$$\boldsymbol{\Lambda}(x) = \prod_{i \in \Psi} (x - \alpha^i), \tag{3.6}$$

is called the error code.

Definition 3.1 was first introduced in [BB13] and from it we conclude the following:

- The product of the polynomials $\boldsymbol{\Lambda}(x)$ and $\boldsymbol{\Gamma}(x)$ has roots at all field elements
$$\boldsymbol{\Lambda}(x)\boldsymbol{\Gamma}(x) = x^n - 1. \tag{3.7}$$
- A codeword from an error code can be expressed as $\boldsymbol{E}(x) = \boldsymbol{\Omega}(x)\boldsymbol{\Gamma}(x)$, where $\boldsymbol{\Omega}(x)$ is the *error evaluator polynomial* with $\deg \boldsymbol{\Omega}(x) < t$.

- The following polynomial multiplication between the error and its locator polynomials:

$$\boldsymbol{\Lambda}(x)\boldsymbol{E}(x) = 0 \mod (x^n - 1). \tag{3.8}$$

- From (3.8), taking only the known part of $\boldsymbol{E}(x)$ into consideration yields [Bos99]:

$$\boldsymbol{\Lambda}(x)\boldsymbol{S}(x) = -\boldsymbol{\Omega}(x) \mod x^{2t}. \tag{3.9}$$

Equations (3.8) and (3.9) are referred to as the key equation.

3.1.1 Error Locator Algorithms

Algorithms used for calculating $\boldsymbol{\Lambda}(x)$ through the key equation are referred to as error locator algorithms in this work. We explain famous examples such as the extended Euclidean [MS88] and Berlekamp–Massey [Ber68] algorithms. These algorithms are able to find $\boldsymbol{\Lambda}(x)$ as long as the number of errors t is smaller than half the minimum distance $\tau = \lfloor (d-1)/2 \rfloor$. We also consider the syndrome extension algorithms, which is also known as power decoding [SSB10]. Unlike the previously mentioned algorithms, this one is able to find $\boldsymbol{\Lambda}(x)$ for a number of errors t which is larger than τ. This is done with the help of some post processing on the received vector $\boldsymbol{r}$. However, this gain comes accompanied by a small probability of failure.

Extended Euclidean Algorithm

Similar to the Euclidean distance, the Euclidean Algorithm [MS88, p. 362] is named after Euclid. Although it is older than a two millennia, it still has many applications in various fields. The main purpose of the algorithm is to find the greatest common divisor (gcd) between two integers. It was later extended to include polynomials and to output more than just their gcd.

Let $\boldsymbol{a}(x), \boldsymbol{b}(x) \in {}^{\mathbb{F}_q[x]}/_{(x^n-1)}$ such that $\deg \boldsymbol{a}(x) < \deg \boldsymbol{b}(x)$, where $\deg(\cdot)$ denotes the degree of a polynomial which is the highest exponent with non-zero coefficient. If $\boldsymbol{b}(x)$ is divided by $\boldsymbol{a}(x)$, a quotient $\boldsymbol{\chi}_1(x) \in {}^{\mathbb{F}_q[x]}/_{(x^n-1)}$ and a remainder $\boldsymbol{\rho}_1(x) \in {}^{\mathbb{F}_q[x]}/_{(x^n-1)}$ are obtained.

$$\boldsymbol{b}(x) = \boldsymbol{a}(x)\boldsymbol{\chi}_1(x) + \boldsymbol{\rho}_1(x). \tag{3.10}$$

The $\mathbf{gcd}(\boldsymbol{a}(x), \boldsymbol{b}(x))$ can be obtained by repeating the process of division in the following manner:

$$\begin{aligned} \boldsymbol{a}(x) &= \boldsymbol{\rho}_1(x)\boldsymbol{\chi}_2(x) + \boldsymbol{\rho}_2(x), \\ \boldsymbol{\rho}_1(x) &= \boldsymbol{\rho}_2(x)\boldsymbol{\chi}_3(x) + \boldsymbol{\rho}_3(x), \\ \boldsymbol{\rho}_2(x) &= \boldsymbol{\rho}_3(x)\boldsymbol{\chi}_4(x) + \boldsymbol{\rho}_4(x), \\ &\vdots \\ \boldsymbol{\rho}_{l-1}(x) &= \boldsymbol{\rho}_l(x)\boldsymbol{\chi}_l(x) + 0. \end{aligned} \tag{3.11}$$

The process ends when the remainder becomes zero and $\mathbf{gcd}(\boldsymbol{a}(x), \boldsymbol{b}(x)) = \boldsymbol{\rho}_l(x)$ is obtained. Equations (3.10) and (3.11) can be extended and written in terms of both $\boldsymbol{a}(x)$ and $\boldsymbol{b}(x)$. This form of the Euclidean Algorithm is called the Extended Euclidean Algorithm (EEA).

$$\begin{aligned} \boldsymbol{b}(x) &= \boldsymbol{a}(x)\boldsymbol{U}_{-1}(x) + \boldsymbol{b}(x)\boldsymbol{V}_{-1}(x), \\ \boldsymbol{a}(x) &= \boldsymbol{a}(x)\boldsymbol{U}_0(x) + \boldsymbol{b}(x)\boldsymbol{V}_0(x), \\ \boldsymbol{\rho}_1(x) &= \boldsymbol{a}(x)\boldsymbol{U}_1(x) + \boldsymbol{b}(x)\boldsymbol{V}_1(x), \\ \boldsymbol{\rho}_2(x) &= \boldsymbol{a}(x)\boldsymbol{U}_2(x) + \boldsymbol{b}(x)\boldsymbol{V}_2(x), \\ &\vdots \\ \boldsymbol{\rho}_l(x) &= \boldsymbol{a}(x)\boldsymbol{U}_l(x) + \boldsymbol{b}(x)\boldsymbol{V}_l(x), \\ \boldsymbol{\rho}_{l+1}(x) = 0 &= \boldsymbol{a}(x)\boldsymbol{U}_{l+1}(x) + \boldsymbol{b}(x)\boldsymbol{V}_{l+1}(x), \end{aligned} \tag{3.12}$$

where $\boldsymbol{U}_i(x)$ and $\boldsymbol{V}_i(x)$ are defined by

$$\begin{aligned} \boldsymbol{U}_i(x) &= \boldsymbol{U}_{i-2}(x) - \boldsymbol{U}_{i-1}(x)\boldsymbol{\chi}_i(x), \\ \boldsymbol{V}_i(x) &= \boldsymbol{V}_{i-2}(x) - \boldsymbol{V}_{i-1}(x)\boldsymbol{\chi}_i(x), \end{aligned} \tag{3.13}$$

with the below initial values

$$\begin{aligned} \boldsymbol{U}_{-1}(x) &= 0, \qquad \boldsymbol{U}_0(x) = 1, \\ \boldsymbol{V}_{-1}(x) &= 1, \qquad \boldsymbol{V}_0(x) = 0. \end{aligned} \tag{3.14}$$

The application of the EEA in decoding of RS codes has two variants depending on the inputs given to the algorithm. Either by using the transform of the received word $\boldsymbol{R}(x)$ (based on Equation (3.8)) or using the syndrome $\boldsymbol{S}(x)$ (based on Equation (3.9)). Here, we only discuss the version where we input $\boldsymbol{R}(x)$, such that $\mathbf{gcd}(\boldsymbol{R}(x), x^n - 1)$ is calculated. However, we are not interested in the gcd itself, but rather the $\boldsymbol{U}$s and $\boldsymbol{V}$s we get when a certain condition is satisfied.

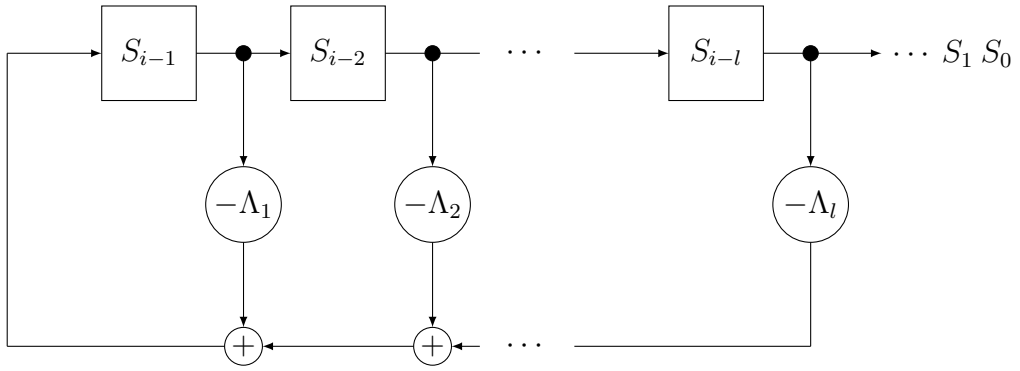

Figure 3.1: Generating the sequence $\boldsymbol{S}$ using a linear feedback shift register.

Theorem 3.1 (Extended Euclidean Algorithm Decoder [Bos99]) *Given $\boldsymbol{R}(x) = \boldsymbol{C}(x) + \boldsymbol{E}(x)$, where $\boldsymbol{E}(x)$ is the DFT of the error polynomial $\boldsymbol{e}(x)$ with $t \leq \tau$ non-zero coefficients. The process of calculating $\mathbf{gcd}(\boldsymbol{R}(x), x^n - 1)$ using the EEA is interrupted when $\deg \boldsymbol{R}_{h-1}(x) > k - 1 + \deg \boldsymbol{U}_{h-1}(x)$ and $\deg \boldsymbol{R}_h(x) \leq k - 1 + \deg \boldsymbol{U}_h(x)$. Then, the following applies:*

$$\boldsymbol{\Lambda}(x) = \boldsymbol{U}_h(x),\ \boldsymbol{\Omega}(x) = -\boldsymbol{V}_h(x). \tag{3.15}$$

The proof of Theorem 3.1 can be found in [Bos99]. The Euclidean algorithm is used later in Chapter 4, where the main focus is the case where the number of errors t is larger than τ. In this case, the error locator polynomial $\boldsymbol{\Lambda}(x)$ is not directly obtained, however, if soft information is available further steps can be done to achieve successful decoding.

Berlekamp–Massey Algorithm

The Berlekamp–Massey Algorithm (BMA) [Ber68, Chapter 7] is named after Elwyn Berlekamp, who introduced the decoding concept and James Massey, who later simplified its description. The main goal of the algorithm is to synthesize the shortest Linear Feedback Shift Register (LFSR) that generates a certain sequence (see Figure 3.1). In our context, the given sequence is the syndrome $\boldsymbol{S}(x)$, and the output should be an LFSR described by $\boldsymbol{\Lambda}(x)$ such that:

$$\Lambda_0 S_i + \Lambda_1 S_{i-1} + \cdots + \Lambda_t S_{i-t} = 0 \quad \forall i = t, \ldots, n - k - 1 \tag{3.16}$$

Note that the linear system of equations (3.16) is directly obtained from the key equation (3.9) by taking the coefficients for degree higher than t.

The process of synthesis starts by assuming that the LFSR is of length one. In each iteration, it is checked whether the LFSR satisfies all equations in (3.16) or not. In case they are not satisfied, the LFSR coefficients are then modified if possible. If no such modification is possible the LFSR is increased in length by one. The process is then repeated until either the LFSR satisfies (3.16) and $\mathbf{\Lambda}(x)$ is found or the decoding fails. Since we have a sequence of length $n-k=d-1$, the longest LFSR which can be obtained is τ. Which means, as long as the number of errors t is smaller τ, an LFSR can always be synthesized. The BMA is shown in Algorithm 3.1.

Algorithm 3.1 BMA Algorithm [Ber68]

Input: $\boldsymbol{S} = (S_0, \ldots, S_{n-k-1})$

Initializations: $l \leftarrow 0$, $\widetilde{l} \leftarrow 0$, $\widetilde{j} \leftarrow -1$
$\mathbf{\Lambda}(x) \leftarrow 1$, $\widetilde{\mathbf{\Lambda}}(x) \leftarrow 1$, $\widetilde{\omega} \leftarrow 1$

1: **for** $j = 1 : n-k-1$ **do**
2: $\quad \omega \leftarrow S_j + \sum_{i=1}^{l} \Lambda_i S_{j-1}$
3: $\quad$ **if** $\omega \neq 0$ **then**
4: $\quad\quad$ **if** $j - \widetilde{j} \leq l - \widetilde{l}$ **then**
5: $\quad\quad\quad \mathbf{\Lambda}(x) \leftarrow \mathbf{\Lambda}(x) - \frac{\omega}{\widetilde{\omega}} \widetilde{\mathbf{\Lambda}}(x) x^{j-\widetilde{j}}$
6: $\quad\quad$ **else**
7: $\quad\quad\quad l^* \leftarrow$, $\mathbf{\Lambda}^*(x) \leftarrow \mathbf{\Lambda}(x)$
8: $\quad\quad\quad \mathbf{\Lambda}(x) \leftarrow \mathbf{\Lambda}(x) - \frac{\omega}{\widetilde{\omega}} \widetilde{\mathbf{\Lambda}}(x) x^{j-\widetilde{j}}$
9: $\quad\quad\quad l \leftarrow j - \widetilde{j} + \widetilde{l}$
10: $\quad\quad\quad \widetilde{l} \leftarrow l^*$, $\widetilde{\mathbf{\Lambda}}(x) \leftarrow \mathbf{\Lambda}^*(x)$
11: $\quad\quad\quad \widetilde{\omega} \leftarrow \omega$, $\widetilde{j} \leftarrow j$
12: $\quad\quad$ **end if**
13: $\quad$ **end if**
14: **end for**

Output: The error locator polynomial $\mathbf{\Lambda}(x)$

Syndrome Extension (Power Decoding)

This decoding method was first introduced in [SSB10]. Its goal is to calculate $\mathbf{\Lambda}(x)$ even when the number of errors t is greater than τ. This is done by calculating extra syndrome equations, by powering the coefficients of the received

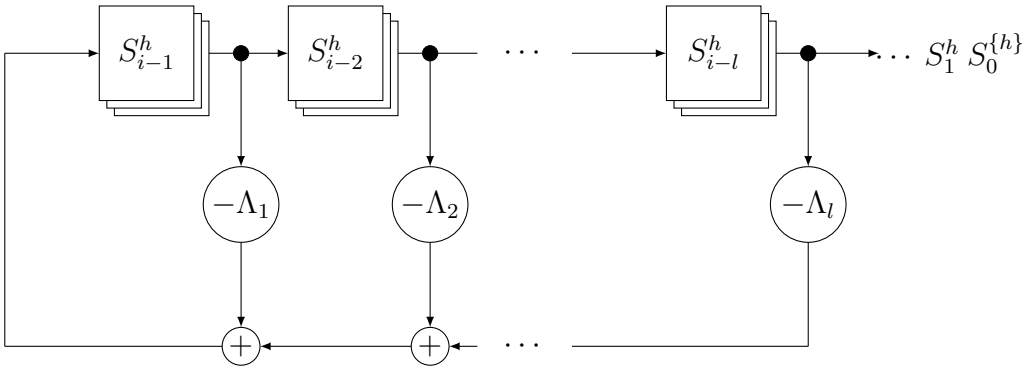

Figure 3.2: Generating the sequences $\boldsymbol{S}^h \, \forall h = 1, \ldots, \ell$ using a linear feedback shift register.

word $\boldsymbol{r}(x)$ with some positive integer ℓ.

$$\begin{aligned} \boldsymbol{r}^\ell(x) :=& \sum_{i=0}^{n-1} r_i^\ell x^i = \sum_{i=0}^{n-1} (c_i + e_i)^\ell x^i = \sum_{i=0}^{n-1} (c_i^\ell + \widetilde{e}_i^{(\ell)}) x^i \\ =& \boldsymbol{c}^\ell(x) + \widetilde{\boldsymbol{e}}^{(\ell)}(x), \end{aligned} \tag{3.17}$$

such that for some i, $e_i = 0$ yields $\widetilde{e}_i^{(\ell)} = 0$, but not necessarily the other way around. Equivalently, the indices of the non-zero coefficients of $\widetilde{\boldsymbol{e}}^{(\ell)}(x)$ are a subset of those of $\boldsymbol{e}(x)$. This is similar to interleaving multiple codewords from different RS codes. However, power decoding is done with only a single low rate RS code. It follows from Definition 2.9 that powering $\boldsymbol{c}(x)$ element-wise to $\boldsymbol{c}^\ell(x)$ is equivalent to powering the polynomial $\boldsymbol{C}(x)$. Since $\deg \boldsymbol{C}(x) \leq k-1$, the relation $\deg (\boldsymbol{C}(x))^\ell \leq \ell(k-1)$ is implied, resulting in $\boldsymbol{c}^\ell(x) \in \mathcal{RS}(n, k^{(\ell)})$, where $k^{(\ell)} := \ell(k-1)+1)$ for all ℓ with $\ell(k-1)+1 \leq n$. We denote the maximum ℓ for which this inequality is fulfilled by ℓ_p.

Having these properties, we are able to build an IRS codeword (see Section 2.4) from a single RS codeword. Since the errors are at the same positions in all vectors $\boldsymbol{r}^\ell(x)$, collaborative decoding can be used to improve the decoding capability. Two examples of collaborative decoding methods are [SSB06] and [Nie16], which are extensions of both the BMA and EEA respectively. Similar to the BMA, the method presented in [SSB06] can be viewed as a multi-sequence shift register synthesis, where we search for the shortest LFSR that generates multiple sequences (see Figure 3.2).

An upper bound for ℓ_p [SSB10, Section 5] is given by

$$\ell_p \leq \frac{\sqrt{(k+3)^2 + 8(k-1)(n-1)} - (k+3)}{2(k-1)}. \tag{3.18}$$

Using powers $\ell \leq \ell_p$ power decoding can correct up to

$$\tau_\ell := \left\lfloor \frac{2\ell n - \ell(\ell+1)k + \ell(\ell-1)}{2(\ell+1)} \right\rfloor \leq \tau_{\ell_p} = \tau_p. \tag{3.19}$$

However, this increase in decoding radius does not come at no cost. In the cases where the powering does not produce new linearly independent equations the decoding fails. The probability of decoding failure is shown to be negligible in [SSB10] and decreases with the increase of the field size q.

3.1.2 Gorenstein–Zierler Error Evaluation Algorithm

Getting to know the number of errors t and where the non-zero coefficients in $\boldsymbol{e}(x)$ is considered the hard part in the decoding process. On the other hand, calculating the error values after that is considered a straight forward operation. An algorithm that is dedicated to calculating the error values is denoted as error evaluator algorithm. Examples of such algorithms are the Gorenstein–Zierler (GZ) [GZ61] and the Forney [For65] algorithms. In subsequent chapters, we use the GZ algorithm as an error evaluator algorithm.

Since $\boldsymbol{e}(x) = \sum_{j\in\Psi} e_j x^j$ and $S_i = E_{i+k} = \boldsymbol{e}(\alpha^{-(i+k)})$, an LSE of equations can be formed:

$$\begin{pmatrix} \alpha^{-j_0 k} & \alpha^{-j_1 k} & \dots & \alpha^{-j_{t-1} k} \\ \alpha^{-j_0 (k+1)} & \alpha^{-j_1 (k+1)} & \dots & \alpha^{-j_{t-1} (k+1)} \\ \vdots & \vdots & \vdots & \vdots \\ \alpha^{-j_0 (n-1)} & \alpha^{-j_1 (n-1)} & \dots & \alpha^{-j_{t-1} (n-1)} \end{pmatrix} \cdot \begin{pmatrix} e_{j_0} \\ e_{j_1} \\ \vdots \\ e_{j_{t-1}} \end{pmatrix} = \begin{pmatrix} S_0 \\ S_1 \\ \vdots \\ S_{n-k-1} \end{pmatrix}, \tag{3.20}$$

where $\Psi = \{j_0, j_1, \dots, j_{t-1}\}$. In (3.20), only $e_{j_0}, \dots, e_{j_{t-1}}$ are unknowns. Thus, only $t < (n-k)$ equations are required and solving the LSE in (3.21) yields the error vector $\boldsymbol{e}$

$$\begin{pmatrix} \alpha^{-j_0 k} & \alpha^{-j_1 k} & \dots & \alpha^{-j_{t-1} k} \\ \alpha^{-j_0 (k+1)} & \alpha^{-j_1 (k+1)} & \dots & \alpha^{-j_{t-1} (k+1)} \\ \vdots & \vdots & \vdots & \vdots \\ \alpha^{-j_0 (k+t-1)} & \alpha^{-j_1 (k+t-1)} & \dots & \alpha^{-j_{t-1} (k+t-1)} \end{pmatrix} \cdot \begin{pmatrix} e_{j_0} \\ e_{j_1} \\ \vdots \\ e_{j_{t-1}} \end{pmatrix} = \begin{pmatrix} S_0 \\ S_1 \\ \vdots \\ S_{t-1} \end{pmatrix}. \tag{3.21}$$

3.2 Interpolation-based Decoding

In syndrome-based decoding, the target is to evaluate the error $\boldsymbol{e}(x)$ then calculate the codeword $\boldsymbol{c}(x)$. However, in interpolation-based decoding, the goal (in most cases) is to directly evaluate $\boldsymbol{C}(x)$ from the received word $\boldsymbol{r}(x)$. The first interpolation-based decoding algorithm is the Welch–Berlekamp algorithm [WB86]. It has a classical decoding radius of half the minimum distance $\tau = \lfloor (d-1)/2 \rfloor$. Later, this work was extended by Madhu Sudan in [Sud97] and a radius beyond τ was obtained. This was done by outputting a list of possible solutions rather than just a unique solution. Thats why usually interpolation-based decoding are referred to as list decoders. List decoders gained a lot of popularity in the last decade and various enhancements and applications for this type of decoding were established. In the same manner as in syndrome-based decoding, the procedure is divided into two steps. First, the process of interpolation in which a bivariate polynomial $\boldsymbol{Q}(x, y)$ is calculated. While satisfying some conditions, $\boldsymbol{Q}(x, y)$ attain certain y-roots which relate to the decoding output. Thats why the second step is called the root finding step. Two of the famous interpolation algorithms are used in this work, namely the Guruswami–Sudan (GS) [GS99] and the Wu [Wu08] algorithms. While for the root finding step, the Roth-Ruckenstein (RR) algorithm [RR00] is used.

3.2.1 Interpolation Algorithms

Guruswami–Sudan Algorithm

The Sudan algorithm [Sud97] has a drawback when high rate codes are used; its decoding radius does not go beyond τ. In order to fix this draw back, the algorithm was extended in [GS99]. At the cost of complexity, the algorithm introduced the idea of multiplicities. A bivariate polynomial $\boldsymbol{Q}(x, y)$ is said to have a zero at some point (a, b) with multiplicity s, if the polynomial $\boldsymbol{Q}^*(x, y) = \boldsymbol{Q}(x + a, y + b) = \sum q^*_{i,j} x^i y^j$ has the coefficients $q^*_{i,j} = 0$ for any $i + j < s$. For a single point (a, b), the total number of coefficients $q^*_{i,j}$ which are equal to zero is $\binom{s+1}{2}$.

Theorem 3.2 (Guruswami–Sudan algorithm [GS99])
For some positive non-zero integers ℓ, s, τ_{GS} and the bivariate non-zero polynomial $\boldsymbol{Q}(x, y)$:

$$\boldsymbol{Q}(x, y) = \boldsymbol{Q}_0(x) + \boldsymbol{Q}_1(x)y + \boldsymbol{Q}_2(x)y^2 + \cdots + \boldsymbol{Q}_\ell(x)y^\ell. \tag{3.22}$$

If $t \leq \tau_{GS}$ and the following conditions are satisfied

1. $\boldsymbol{Q}(\alpha^i, r_i) = 0$ *with multiplicity* $s, \forall i = 0, \ldots, n-1$

2. $\deg \boldsymbol{Q}_i(x) \leq s(n - \tau_{GS}) - i(k-1) - 1, \forall i = 0, \ldots, \ell$

then $(y - \boldsymbol{C}(x))$ *divides* $\boldsymbol{Q}(x, y)$.

The list size ℓ, multiplicity s are chosen depending on the required decoding radius τ_{GS}. The cases where $s = 1$ and $\ell = s = 1$ correspond to the Sudan [Sud97] and Welch–Berlekamp [WB86] algorithms respectively. From the degree condition, the number of unknown coefficients in $\boldsymbol{Q}(x, y)$ is

$$\begin{aligned} N_u &= \sum_{i=0}^{\ell} s(n - \tau_{GS}) - i(k-1) - 1 \\ &= s(\ell+1)(n - \tau_{GS}) - \frac{1}{2}\ell(\ell+1)(k-1) - 1. \end{aligned} \tag{3.23}$$

Since the polynomial $\boldsymbol{Q}^*(x, y)$ has $\binom{s+1}{2}$ zero coefficients, for n points, we obtain $N_e = n\binom{s+1}{2}$ equations. For $\boldsymbol{Q}(x, y)$ to be non-zero, the number of unknowns should be greater than the number of equations ($N_u > N_e$) as follows:

$$s(\ell+1)(n - \tau_{GS}) - \frac{1}{2}\ell(\ell+1)(k-1) - 1 > n\binom{s+1}{2}, \tag{3.24}$$

A bound on the decoding radius τ_{GS} for a given set of parameters is determined as follows:

$$\tau_{GS} < \frac{n(2\ell - s + 1)}{2(\ell+1)} - \frac{l(k-1)}{2s}. \tag{3.25}$$

Asymptotically, it was shown that the GS algorithm can reach the Johnson bound [GS99] such that

$$\tau_{GS} < n - \sqrt{n(n-d)}. \tag{3.26}$$

The GS algorithm was later extended in [KV03] where the Kötter–Vardy (KV) algorithm was proposed. The KV algorithm enabled soft decoding such that the multiplicity of each point reflects its reliability. Having reliability information, the KV algorithm outperforms the GS algorithm in terms of error correction. However, this comes at the huge cost of complexity as it is known to be a powerful but slow algorithm.

Wu Algorithm

The Wu algorithm is proposed in [Wu08]. Similar to the GS algorithm, interpolation is done and a bivariate polynomial $\boldsymbol{Q}(x, y)$ is calculated. The decoding radius τ_{Wu} is also able to reach the Johnson bound. However, since it uses a different set of points for the interpolation, the y-roots are not related to $\boldsymbol{C}(x)$, but rather to the error locator polynomial $\boldsymbol{\Lambda}(x)$. In the case where the number of errors t is beyond half the minimum distance τ, algorithms like the EEA and BMA usually fail (unless wrong decoding occurs). It turns out that such algorithms do not entirely fail, but rather lack the resources to go further. The Wu algorithms continues where they left off, using their last result as starting point.

Let the coprime polynomials $\boldsymbol{U}(x)$ and $\boldsymbol{V}(x)$ be outputs of the EEA at a certain iteration. It is shown in [BB08] that a basis for $\boldsymbol{\Lambda}(x)$ is given as follows:

$$\boldsymbol{\Lambda}(x) = \boldsymbol{A}(x)\boldsymbol{U}(x) + \boldsymbol{B}(x)\boldsymbol{V}(x), \tag{3.27}$$

where $\boldsymbol{A}(x)$ and $\boldsymbol{B}(x)$ are unknown polynomials satisfying

$$\begin{aligned} \deg \boldsymbol{A}(x) &= \theta_A = \deg \boldsymbol{\Lambda}(x) - \deg \boldsymbol{U}(x), \\ \deg \boldsymbol{B}(x) &= \theta_B \leq \deg \boldsymbol{\Lambda}(x) - d - \deg \boldsymbol{U}(x). \end{aligned} \tag{3.28}$$

From (3.6), we know that $\boldsymbol{\Lambda}(x)$ evaluates to zero at all the error locations $i \in \Psi$:

$$\boldsymbol{A}(\alpha^i)\boldsymbol{U}(\alpha^i) + \boldsymbol{B}(\alpha^i)\boldsymbol{V}(\alpha^i) = 0. \tag{3.29}$$

Equation (3.29) can be written as follows:

$$\beta_i = -\frac{\boldsymbol{V}(\alpha^i)}{\boldsymbol{U}(\alpha^i)} = \frac{\boldsymbol{A}(\alpha^i)}{\boldsymbol{B}(\alpha^i)}. \tag{3.30}$$

In the same manner as in GS, interpolation is generalized to provide a bivariate polynomial $\boldsymbol{Q}(x, y)$ passing through the points (α^i, β_i) for $i = 0, \ldots, n-1$. However, the desired y-root of $\boldsymbol{Q}(x, y)$ is in this case the fraction $\frac{\boldsymbol{B}(x)}{\boldsymbol{A}(x)}$ (not $\boldsymbol{C}(x)$ as in GS), enabling us to calculate $\boldsymbol{\Lambda}(x)$ using (3.29). Since we are looking for a fraction of two polynomials, the interpolation is known as rational interpolation.

Before going into the details of the Wu algorithm, we first describe the general rational interpolation problem and its solution. The goal can be expressed as follows: given points (x_i, β_i) for $i = 0, \ldots, n-1$, one seeks two polynomials $\boldsymbol{f}_1(x)$, $\boldsymbol{f}_2(x) \in \mathbb{F}_q(x)$ and $\frac{\boldsymbol{f}_1(x_i)}{\boldsymbol{f}_2(x_i)} = \beta_i$ holds for many, but not necessarily all, values of i, and such that $\deg \boldsymbol{f}_1(x)$ and $\deg \boldsymbol{f}_2(x)$ are both small. The fact that $\boldsymbol{f}_2(x)$ might

have roots at some x_i is problematic. Trifonov [Tri10] suggested to consider the β_i as partially projective points $(y_i : z_i) \in \mathbb{P}^1_{\mathbb{F}_q}$, i.e. that $(y_i : z_i) = (\gamma y_i : \gamma z_i)$ for all $\gamma \in \mathbb{F}_q^\star$ and where $(0 : 0)$ is disallowed.

The problem is solved by constructing a polynomial $\boldsymbol{Q} \in \mathbb{F}_q[x][y, z]$, homogeneous in y and z, in such a way that it is guaranteed that $\boldsymbol{Q}(x, \boldsymbol{f}_1(x), \boldsymbol{f}_2(x)) = 0$. The $\boldsymbol{f}_1(x), \boldsymbol{f}_2(x)$ can then be extracted from $\boldsymbol{Q}$ as roots, which is possible since y and z are homogeneous in $\boldsymbol{Q}$. The following theorem is a rephrasing of [Tri10, Lemma 3]:

Theorem 3.3 (Rational Interpolation)
Let ℓ, s and T be positive integers, and let the points $\{(x_0, y_0 : z_0), (x_1, y_1 : z_1), \ldots, (x_{N-1}, y_{N-1} : z_{N-1})\}$ be $N \geq T$ points in $\mathbb{F}_q \times \mathbb{P}^1_{\mathbb{F}_q}$. Assume that $\boldsymbol{Q}(x, y, z) = \sum_{i=0}^{\ell} \boldsymbol{Q}(x) y^i z^{\ell - i}$ is non-zero and such that $(x_i, y_i : z_i)$ are zeroes of multiplicity s for all $i = 0, \ldots, n-1$, and $\deg_{(1, w_1, w_2)} \boldsymbol{Q}(x, y, z) < sT$, for two $w_1, w_2 \in \mathbb{R}_+ \cup \{0\}$. Any two coprime polynomials $\boldsymbol{f}_1(x), \boldsymbol{f}_2(x)$ satisfying $\deg \boldsymbol{f}_1(x) \leq w_1, \deg \boldsymbol{f}_2(x) \leq w_2$, as well as, $z_i \boldsymbol{f}_1(x_i) + y_i \boldsymbol{f}_2(x_i) = 0$ for at least T values of i, will satisfy $\boldsymbol{Q}(x, \boldsymbol{f}_1(x), \boldsymbol{f}_2(x)) = 0$.

In the above, $\deg_{(w_x, w_y, w_z)}$ is the (w_x, w_y, w_z)-weighted degree, such that

$$\deg_{(w_x, w_y, w_z)} x^i y^j z^h = w_x i + w_y j + w_z h. \tag{3.31}$$

For polynomials, the weighted degree is the maximal weighted degree of its monomials. The list size ℓ and multiplicity s, are not part of the rational interpolation problem one wishes to solve. They should nevertheless be chosen in such a way to make it possible to construct the $\boldsymbol{Q}$-polynomial. One can regard the root-requirements for $\boldsymbol{Q}$ as a linear system of equations in the monomials of $\boldsymbol{Q}$, and the weighted degree constraints as a bound on the number of monomials available. This gives a bound on the parameters on when it is guaranteed that a satisfactory $\boldsymbol{Q}$ exists. Further analysis shows that this is the case if the following is satisfied:

$$T^2 > N(w_1 + w_2). \tag{3.32}$$

Such an analysis along with precise choices of ℓ and s can be found in [Tri10] or in more detail in [Nie13, Proposition 5.7].

Making use of rational interpolation tool from Theorem 3.3 with the number of points $N = n$ and plugging our polynomials $\boldsymbol{U}(x)$ and $\boldsymbol{V}(x)$, we get the Wu algorithm:

Theorem 3.4 (Wu algorithm)
For positive non-zero integers ℓ, s, τ_{Wu} and a trivariate non-zero polynomial $\boldsymbol{Q}(x, y, z)$:

$$\boldsymbol{Q}(x, y, z) = \boldsymbol{Q}_0(x)z^{\ell} + \boldsymbol{Q}_1(x)yz^{\ell-1} + \boldsymbol{Q}_2(x)y^2z^{\ell-2} + \cdots + \boldsymbol{Q}_{\ell}(x)y^{\ell}. \quad (3.33)$$

If $t \leq \tau_{Wu}$ and the following conditions are satisfied

1. $\boldsymbol{Q}(\alpha^i, -V(\alpha^i), U(\alpha^i)) = 0$ *with multiplicity* $s, \forall i = 0, \ldots, n-1$
2. $\deg \boldsymbol{Q}_i(x) \leq s\tau_{Wu} - (\ell - i)\theta_A - i\theta_B - 1, \forall i = 0, \ldots, \ell$

then $(\boldsymbol{B}(x)y - \boldsymbol{A}(x)z)$ divides $\boldsymbol{Q}(x, y, z)$.

As in GS, the choice of ℓ and s, depends on the desired correction radius τ_{Wu}. In order for the interpolation to succeed, the number of unknowns should be more than the number of equations. Thus, the following inequality must be satisfied:

$$(\ell + 1)s\tau_{Wu} - \binom{\ell + 1}{2}(\theta_A + \theta_B) > \binom{s + 1}{2}n. \quad (3.34)$$

In [Wu08], it is shown that the Wu algorithm can decode also up to the Johnson bound as the GS algorithm

$$\tau_{Wu} < n - \sqrt{n(n - d)}. \quad (3.35)$$

3.2.2 Root Finding using Roth–Ruckenstein Algorithm

Regardless of which interpolation method is used, the process of decoding can not be finished without extracting the factor $(y - \boldsymbol{C}(x))$ from $\boldsymbol{Q}(x, y)$ if GS was used or $(\boldsymbol{B}(x)y - \boldsymbol{A}(x)z)$ from $\boldsymbol{Q}(x, y, z)$ in case of Wu. The Roth–Ruckenstein (RR) algorithm is known to be one of the efficient root finding algorithms [RR00], [Rot06, Page 284]. Here, Algorithm 3.2 only shows the case of factorizing a bivariate polynomial $\boldsymbol{Q}(x, y)$. For the case of the trivariate polynomials, more details can be found in [Nie13, Section 5.1].

As shown in Algorithm 3.2, the RR algorithm tries to find the polynomial $\boldsymbol{f}(x) = f_0 + \cdots + f_{k-1}x^{k-1}$ such that $(y - \boldsymbol{f}(x))$ divides $\boldsymbol{Q}(x, y)$. In the first run, f_0 is calculated and a new bivariate polynomial $\boldsymbol{T}(x, y)$ is obtained with a y-root at $\widetilde{\boldsymbol{f}}(x) = \boldsymbol{f}(x) - f_0$. The algorithm takes $\boldsymbol{T}(x, y)$ as input to a second run and recursively calculates $\widetilde{f_0} = f_1$. The process is repeated, until all coefficients are calculated. The output of the algorithm is a list $\mathcal{F}$ containing all y-roots of $\boldsymbol{Q}(x, y)$.

Algorithm 3.2 Root-finding algorithm (RR) [RR00, Rot06]

Input: Bivariate polynomial $\boldsymbol{Q}(x, y)$, dimension k, and $h \in \mathbb{N}$
Global Variables: Set $\mathcal{L} \subseteq \mathbb{F}[x]$
Polynomial $\boldsymbol{f}(x) \in \mathbb{F}[x]$

1: **if** $h = 0$ **then**
2: $\quad \mathcal{U} = \emptyset$
3: **end if**
4: $m \leftarrow$ largest integer such that x^m divides $\boldsymbol{Q}(x, y)$
5: $\boldsymbol{T}(x, y) \leftarrow x^{-m}\boldsymbol{Q}(x, y)$
6: $\mathcal{Z} \leftarrow$ set of all distinct y-roots of $\boldsymbol{T}(0, y)$ in $\mathbb{F}$
7: **for** each $\gamma \in \mathcal{Z}$ **do**
8: $\quad f_h \leftarrow \gamma$
9: $\quad$ **if** $h < k - 1$ **then**
10: $\quad\quad RR(\boldsymbol{T}(x, xy + \gamma), k, h + 1)$
11: $\quad$ **else**
12: $\quad\quad$ **if** $\boldsymbol{Q}(x, \boldsymbol{f}(x)) = 0$ **then**
13: $\quad\quad\quad \mathcal{L} \leftarrow \mathcal{L} \cup \{\boldsymbol{f}(x)\}$
14: $\quad\quad$ **end if**
15: $\quad$ **end if**
16: **end for**

Output: The list of y-roots $\mathcal{L}$

3.3 Error/Erasure Decoding

3.3.1 Definition of an Erasure

Erasing a coordinate in the received vector $\boldsymbol{r}$ is simply removing its effect from the decoding process. There are many ways to describe an erasure. In some systems, it can happen that the channel would produce a received vector $\boldsymbol{r}$ with a coefficient $r_i \notin \mathbb{F}$. This value is then considered corrupt and its position is replaced by an erasure. In this dissertation, an erasure is viewed in another way. Whenever the location of the error becomes known, it can be replaced by an erasure. Knowing where errors are located is not always possible, but sometimes it is possible to obtain some reliability information. Calculating reliability information is explained in details in Section 4.1. A coordinate with low reliability is then considered an erasure. Of course mistakenly erasing a non-erroneous coordinate will reduce the decoding capabilities. However, the gain from erasing unreliable coordinates is greater than its loss as shown later in Chapter 4. Let

$\phi \in \{0, \ldots, n-1\}$ be the set of coordinates where an erasure is introduced. The number of erasures is denoted as $\varepsilon = \#\phi$.

Erasures in syndrome decoding

In classical decoding techniques such as EEA and BMA (see Section 3.1), dealing with error and erasures transforms (3.1) to the following form:

$$\tau = \lfloor (d - \varepsilon - 1)/2 \rfloor. \tag{3.36}$$

In (3.36), the errors in the non-erased part of $\boldsymbol{r}$ are denoted τ. If all error locations become known, $\varepsilon \leq d-1$ erasures can be corrected. It is because of this, it can be said that errors are twice as expensive as erasures. Let $\boldsymbol{\Phi}(x)$ be the erasure locator polynomial such that

$$\boldsymbol{\Phi}(x) = \prod_{i \in \phi} (x - \alpha^i). \tag{3.37}$$

Only the input of the decoding processes of EEA and BMA changes when decoding erasures along with errors. For EEA (Theorem 3.1), instead of using $\boldsymbol{R}(x)$ as input, we use $\widehat{\boldsymbol{R}}(x) = \boldsymbol{\Phi}(x)\boldsymbol{R}(x) \mod (x^n - 1)$. While in BMA (Equation (3.16)), the input sequence $\widehat{\boldsymbol{S}}(x) = \widetilde{S}_\varepsilon + \widetilde{S}_{\varepsilon+1}x \cdots \widetilde{S}_{d-2}x^{d-\varepsilon-2}$ is used, where $\widetilde{\boldsymbol{S}}(x) = \boldsymbol{\Phi}(x)\boldsymbol{S}(x) \mod x^{d-1}$.

Erasures in interpolation decoding

Introducing erasures in interpolation decoding methods like GS is simpler. The erased positions should just be ignored during the interpolation step. The points (α^i, r_i) with $i \in \phi$ should be excluded from Theorem 3.2. The parameters ℓ, s and τ_{GS} however will now depend on the new number of interpolation points $\widehat{n} = n - \varepsilon$. They should chosen such that (3.25) remains satisfied.

In Wu, introducing erasures does not make sense, since the goal of the interpolation is to get the polynomial $\boldsymbol{Q}(x, y, z)$ passing through erroneous positions. It is, however, useful to exclude non-erroneous positions from the interpolation. This is shown later in Chapter 4.

3.3.2 Generalized Minimum Distance Decoding

Generalized Minimum Distance (GMD) decoding was introduced by Forney in [For66]. It is a decoding method that is able to combine reliability information

and error/erasure decoding. Later on, variants of GMD decoding started to appear while keeping the name and origin. GMD can be used for either decoding with reliability or decoding concatenated codes as in [Bos99, Chapter 7] and [Rot06, Chapter 12]. The variant we are using is analogous to the one in [Bos99] and is shown in Algorithm 3.3.

Before explaining GMD decoding, we first illustrate how reliability information is provided to the decoder. Let the reliability information be represented by the vector $\boldsymbol{\eta} = (\eta_0, \ldots, \eta_{n-1})$. Each element in η_i reflects the probability that a position i is erroneous $\big(P(i \in \Psi)\big)$ for all $i \in \{0, \ldots, n-1\}$. For example, the value in η_i can be inversely proportional with the probability $P(i \in \Psi)$. The lower the value η_i, the more probable that the coordinate i is erroneous. There is however no standard way on how these two values are related and is totally dependant on the system at hand.

Algorithm 3.3 Generalized Minimum Distance (GMD) decoding

Input: Received vector $\boldsymbol{r}$, minimum distance d, reliability vector $\boldsymbol{\eta}$ and an error/erasure decoder $\boldsymbol{\Xi}$

Initializations: $\varepsilon \leftarrow 0$, $\mathcal{L} \leftarrow \emptyset$

1: **while** $\varepsilon < d$ **do**
2: $\quad \widetilde{\boldsymbol{r}} \leftarrow$ Erase the most ε unreliable positions in $\boldsymbol{r}$ depending on $\boldsymbol{\eta}$
3: $\quad \widetilde{\boldsymbol{c}} \leftarrow$ Decode $\widetilde{\boldsymbol{r}}$ using decoder $\boldsymbol{\Xi}$
4: $\quad$ **if** Decoding success **then**
5: $\quad\quad \mathcal{L} = \mathcal{L} \cup \widetilde{\boldsymbol{c}}$
6: $\quad$ **end if**
7: $\quad \varepsilon \leftarrow \varepsilon + 2$
8: **end while**
9: $\widehat{\boldsymbol{c}} \leftarrow \underset{\boldsymbol{l} \in \mathcal{L}}{\operatorname{argmin}} \, d_{\mathrm{G}}(\boldsymbol{l}, \boldsymbol{r})$

Output: The codeword $\widehat{\boldsymbol{c}}$

GMD decoding is a multi trial decoding procedure, where multiple decoding attempts using any error/erasure decoder $\boldsymbol{\Xi}$ are excuted. In the beginning, the decoder $\boldsymbol{\Xi}$ attempts to decode the received vector $\boldsymbol{r}$ without introducing any erasures. If a decoding result exists, it is saved in a list $\mathcal{L}$. In the second attempt, two erasures are introduced to the received vector $\boldsymbol{r}$ and decoding is repeated. The erasures are introduced at the most unreliable positions in $\boldsymbol{r}$ depending on the reliability vector $\boldsymbol{\eta}$. If the second attempt results in a new decoding result, it is added to the list $\mathcal{L}$. The process is then repeated with

adding two new erasures with every new decoding attempt. Depending on some defined criteria, a unique solution is chosen from the list $\mathcal{L}$ and is considered the output of the decoder. In [For66], the output codeword is the code word having the smallest *generalized distance* $d_{\mathrm{G}}(\boldsymbol{r}, \widehat{\boldsymbol{c}})$.

$$d_{\mathrm{G}}(\widehat{\boldsymbol{c}}, \boldsymbol{r}) = \sum_{i=0}^{n-1} d_g(\widehat{c}_i, r_i), \tag{3.38}$$

where

$$d_g(\widehat{c}_i, r_i) = \begin{cases} \delta_{g_i}, & \widehat{c}_i = r_i, \\ 1 - \delta_{g_i}, & \widehat{c}_i \neq r_i. \end{cases} \tag{3.39}$$

The value of $\delta_{g_i} \in [0, 1/2]$ is calculated depending on the reliability information η_i, such that $\delta_{g_i} = 0$ for the most reliable positions and $\delta_{g_i} = 1/2$ for most unreliable positions.

In the remaining chapters, we do not use the generalized distance defined by (3.38) and was only mentioned for completeness. Also, GMD decoding is only used as a performance reference for other algorithms in this dissertation. However, since combining reliability information with error/erasure decoding is the main building block of this dissertation, we consider it as the inspiration for most the new algorithms to be presented later in Chapters 4 and 5.

4

Decoding with Reliability in Finite Fields

When it comes to error correction, the optimal decoding method is maximum likelihood decoding [MS88], where the probability of having wrong decoding decision is minimized. However, using this method is computationally expensive as it has exponential complexity. So suboptimal decoding methods like minimum distance decoding are used since they provide acceptable error correction capabilities while having a reasonable complexity. Examples of algorithms in this category are those presented in Chapter 3. In order to increase the correction radius and provide a performance close to that of maximum like decoding, we can use reliability information. Reliability is a form of soft information. It is extra information given to a decoder to produce a better decoding performance. Depending on the communications system, it can be either given by an outside source "genie" or can be calculated at the receiver. Different algorithms use the reliability in different ways. For binary codes, the Chase [Cha72] algorithm flips the positions with low reliability, while the Dorsch [Dor74] algorithm focuses more on recovery using positions with high reliability. For non-binary codes, GMD decoding [For66] introduces erasures at positions with low reliability. There is also the Kötter–Vardy (KV) algorithm [KV03] which is an extension to the GS algorithm. In KV, the reliability determines the multiplicity of the points used in the interpolation. The higher the reliability for a position, the higher the multiplicity assigned to its corresponding interpolation point. The KV algorithm is considered to give one of the best performances when compared to other RS decoding techniques that use reliabilities. Therefore, we use the KV algorithm as a reference algorithm for comparison. Before going into details of the proposed algorithms, we first show the different ways to calculate reliabilities.

4.1 Reliability Calculation

A received vector $\boldsymbol{r} \in \mathbb{F}_q^n$ is coupled with a reliability vector $\boldsymbol{\eta} \in \mathbb{R}^n$. The elements of $\boldsymbol{\eta} = (\eta_0, \ldots, \eta_{n-1})$ indicate if a position of $\boldsymbol{r}$ is likely to be erroneous, in other words, if it is reliable or not. The reliability η_i is inversely proportional to the probability of position i being in the error set Ψ $\big(\mathrm{P}(i \in \Psi)\big)$. The lower the value η_i, the higher the probability that the position i is erroneous. Such positions, with low reliability, are called *unreliable* positions for the rest of this chapter. There are various ways of calculating this vector, they all depend on the used channel model. In the following we explain two possible methods.

4.1.1 Code Concatenation

This method of reliability calculation is used in many publications including [SSBZ10] and [CSS14]. If a q-ary symmetric channel exists, a code concatenation is best suited to calculate reliabilities. Code concatenation is initially investigated in [For66], where two codes are being used to create a stronger code. The first code $\mathcal{C}^{\mathsf{o}}(n^{\mathsf{o}}, k^{\mathsf{o}}, d^{\mathsf{o}})$ is called the outer code while the second $\mathcal{C}^{\mathsf{i}}(n^{\mathsf{i}}, k^{\mathsf{i}}, d^{\mathsf{i}})$ is called the inner code. The resulting code will have the following parameters $\mathcal{C}_{\mathsf{c}}(n^{\mathsf{o}}n^{\mathsf{i}}, k^{\mathsf{o}}k^{\mathsf{i}}, d_{\mathsf{c}} \geq d^{\mathsf{o}}d^{\mathsf{i}})$. The encoding process is done such that the information symbols $\boldsymbol{u} \in \mathbb{F}_q^{k^{\mathsf{o}}}$ is mapped to a codeword $\boldsymbol{c}^{\mathsf{o}} \in \mathcal{C}^{\mathsf{o}}$. Each symbol $c_j^{\mathsf{o}} \in \mathbb{F}_q^{k^{\mathsf{i}}}$ is then used as information symbols to the outer encoder and is encoded to form the codeword $\boldsymbol{c}^{\mathsf{i}} \in \mathcal{C}^{\mathsf{i}}$ with symbols $\boldsymbol{c}_j^{\mathsf{i}} \in \mathbb{F}_q$. Errors are introduced by the channel such that the vector $\boldsymbol{r}^{\mathsf{i}} = \boldsymbol{c}^{\mathsf{i}} + \boldsymbol{e}$ is received. The inner decoder either calculates an estimate of the transmitted codeword $\widetilde{\boldsymbol{c}}^{\mathsf{i}}$ or fails. This provides the outer decoder with the vector $\boldsymbol{\Delta} = (\Delta_0, \ldots, \Delta_{n^{\mathsf{o}}-1})$ such that:

$$\Delta_j = \begin{cases} d^{\mathsf{i}}/2, & \text{decoding failure,} \\ d_{\mathrm{H}}(\boldsymbol{r}_j^{\mathsf{i}}, \widetilde{\boldsymbol{c}}_j^{\mathsf{i}}), & \text{otherwise.} \end{cases} \tag{4.1}$$

In this case, the reliability vector $\boldsymbol{\eta}$ is then defined as follows:

$$\eta_j \coloneqq \frac{d^{\mathsf{i}} - 2\Delta_j}{d^{\mathsf{i}}}, \text{ where } \eta_j \in [0, 1]. \tag{4.2}$$

More details and insights on code concatenation and decoding them using reliability information can be found in [Bos99, Chapter 9] and [Rot06, Chapter 12].

4.1.2 Modulation-based

For memoryless Additive White Gaussian Noise (AWGN) channels, we calculate the reliability vector $\boldsymbol{\eta}$ using the method mentioned in [KNIH94, KWB10]. Unlike the previous method, only a single RS code over $\mathbb{F}_{q^m}$ is needed. Each coefficient in a codeword $\boldsymbol{c} \in \mathcal{RS}$ can be written as m elements from $\mathbb{F}_q$ such that $c_i = (c_{i,0}, c_{i,1}, \ldots, c_{i,m-1}) \in \mathbb{F}_q^m$. Using q-ASK modulation, the codeword $\boldsymbol{c}$ is mapped to the vector $\boldsymbol{x} \in \mathbb{R}^{nm}$. It is also possible to use q-PSK, however, for $q > 2$ the elements of the vector $\boldsymbol{x}$ will be complex-valued. In this chapter, we only consider the real- valued case. Through the channel, vector $\boldsymbol{x}$ is transmitted and $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{\nu}$ is received, where $\boldsymbol{\nu}$ is the added noise vector. The elements of $\boldsymbol{\nu}$ are extracted from a normally distributed random variable with mean zero and standard deviation σ. The value of σ depends on the desired Signal-to-Noise Ratio (SNR) such that

$$\mathrm{SNR} = 10 \log_{10} \frac{\mathbb{E}_s}{R \cdot \sigma^2}, \tag{4.3}$$

where $\mathbb{E}_s$ is the energy of a transmitted symbol and $R = k/n$ is the code rate. Let $\boldsymbol{\Delta}$ be a matrix with dimensions $n \times q^m$. An element $\Delta_{i,z} \in \mathbb{R}$ reflects the probability that r_i was a certain element $z \in \mathbb{F}_{q^m}$ before the noise was added

$$\Delta_{i,z} = \ln \frac{\mathrm{P}(r_i | c_i = z)}{\sum\limits_{l \in \mathbb{F}_q^m,\ l \neq z} \mathrm{P}(r_i | c_i = l)}. \tag{4.4}$$

The symbols r_i and z can be written as m elements from $\mathbb{F}_q$ such that $r_i = (r_{i,0}, r_{i,1}, \cdots, r_{i,m-1})$ and $z = (z_0, z_1, \ldots, z_{m-1})$. Assuming the channel is memoryless and the components of the noise vector $\boldsymbol{\nu}$ are drawn from a normal distributed random source with zero mean and standard deviation σ, the following applies

$$\mathrm{P}(r_i | c_i = z) = \prod_{\kappa=0}^{m-1} \mathrm{P}(r_{i,\kappa} | c_{i,\kappa} = z_\kappa), \tag{4.5}$$

$$= \frac{1}{\sqrt{2\pi\sigma^2}^m} \exp\left\{ \sum_{\kappa=0}^{m-1} -\frac{1}{2\sigma^2} (r_{i,\kappa} - z_\kappa)^2 \right\}. \tag{4.6}$$

Combining (4.4) and (4.6) yields

$$\Delta_{i,z} = \ln \frac{\prod_{\kappa=0}^{m-1} \mathrm{P}(r_{i,\kappa}|c_{i,\kappa} = z_\kappa)}{\sum_{l \in \mathbb{F}_q^m,\ l \neq z} \prod_{\kappa=0}^{m-1} \mathrm{P}(r_{i,\kappa}|c_{i,\kappa} = l_\kappa)}. \tag{4.7}$$

$$= \ln \frac{\exp\left\{\sum_{\kappa=0}^{m-1} -\frac{1}{2\sigma^2}(r_{i,\kappa} - z_\kappa)^2\right\}}{\sum_{l \in \mathbb{F}_q^m,\ l \neq z} \exp\left\{\sum_{\kappa=0}^{m-1} -\frac{1}{2\sigma^2}(r_{i,\kappa} - l_\kappa)^2\right\}}. \tag{4.8}$$

After calculating $\boldsymbol{\Delta}$ from (4.8), it is used to define the reliability vector $\boldsymbol{\eta}$ for this case such that

$$\eta_i \coloneqq (\Delta_{i_{1\mathrm{st}}} - \Delta_{i_{2\mathrm{nd}}}), \quad \text{for } i \in 0, \ldots, n-1. \tag{4.9}$$

where $\Delta_{i_{1\mathrm{st}}}$ and $\Delta_{i_{2\mathrm{nd}}}$ are the largest two value in the column Δ_i. The process of calculating $\boldsymbol{\Delta}$ and finding the largest entries in each of its columns increases exponentially with the increase of m. This is because of the size of $\boldsymbol{\Delta}$ with a number of rows equal to the size of the field (q^m).

4.2 Decoding Algorithms Using Reliability

Having calculated the reliability vector $\boldsymbol{\eta}$, we are able to use this extra information to correct more errors. Still, classical methods like BMA or GS are not designed to make use of the reliability information provided. Either some tweaking or extra steps are required to achieve this gain in decoding radius. Similar to the decoding algorithms mentioned in Chapter 3, the algorithms presented in the following sections are two examples of both syndrome-based decoding (Chase-like decoder 4.2.1) and interpolation-decoding (Reduced list-decoding 4.2.2). It is important to note that we are only interested in the cases where the number of errors t exceeds half the minimum distance τ. Using the reliability information, we show that we are able to correct errors beyond τ using the proposed algorithms.

4.2.1 Chase-like Decoder

There exist many Chase-like decoders in the scientific community. They are all alterations of the original Chase decoder presented in [Cha72]. The extent of changes from the original usually depends on the system in which it is implemented and the code to be used in this system. The variant we consider here in this section is the one presented in [MB15]. But before we go into its details, we briefly explain the idea behind the original version of the decoder. In addition, we present a method of the decoding beyond half the minimum distance τ when some side information about the error locations exists.

The original Chase decoder was proposed for binary codes. Given the reliability vector $\boldsymbol{\eta}$, let the set $\mathcal{L} \subset \{0, \ldots, n-1\}$ contain the $L = \#\mathcal{L}$ least reliable positions. Based on some criteria, let the set of some combinations of the elements of $\mathcal{L}$ be $\Upsilon \subset \mathcal{P}(\mathcal{L})$, where $\mathcal{P}(\mathcal{L})$ is the power set of $\mathcal{L}$. For a combination $\upsilon \in \Upsilon$, the corresponding bits are flipped (zeros are changed to ones, while ones are changed to zeros) and the modified received vector is decoded. This process is repeated for all combinations Υ. For q-ary codes, the act of flipping corresponds trying out all the $q-1$ other symbols. For a large q, this is considered tedious work. So instead of flipping, we introduce erasures, which is similar to GMD decoding (see Section 3.3.2).

Beyond Half the Minimum Distance Decoding Using EEA

In the presented Chase-like decoder, error/erasure decoding is done using the following EEA based decoder. If the number of errors t is smaller than or equal to the EEA decoding radius τ, the EEA is able to find $\boldsymbol{\Lambda}(x)$ without going through any extra steps of decoding (see Section 3.1.1). If $t > \tau$, EEA fails in calculating $\boldsymbol{\Lambda}(x)$ but it provides us with coprime polynomials $\boldsymbol{U}(x)$ and $\boldsymbol{V}(x)$ as outputs [BB08]. The degrees of both polynomials are τ and $\tau - 1$ respectively. Let the number of errors be $t = \tau + t_0$. As mentioned before in Section 3.2.1, [BB08] shows that a basis for $\boldsymbol{\Lambda}(x)$ is given as follows:

$$\boldsymbol{\Lambda}(x) = \boldsymbol{A}(x)\boldsymbol{U}(x) + \boldsymbol{B}(x)\boldsymbol{V}(x), \tag{4.10}$$

where $\boldsymbol{A}(x)$ and $\boldsymbol{B}(x)$ are unknown polynomials with degrees:

$$\begin{aligned} \deg \boldsymbol{A}(x) &= t_0 \\ \deg \boldsymbol{B}(x) &\leq t_0 - 1 \end{aligned} \tag{4.11}$$

From (4.11), the number of unknown coefficients in $\boldsymbol{A}(x)$ and $\boldsymbol{B}(x)$ is $2t_0 + 1$. Since we are only interested in the roots of $\boldsymbol{\Lambda}(x)$, we can set any coefficient to

any value (we set $A_{t_0} = 1$). Thus, the number of unknowns becomes $2t_0$. From Definition 3.1, we know that

$$\boldsymbol{\Lambda}(\alpha^i) = \boldsymbol{A}(\alpha^i)\boldsymbol{U}(\alpha^i) + \boldsymbol{B}(\alpha^i)\boldsymbol{V}(\alpha^i) = 0, \quad \text{where } i \in \Psi. \tag{4.12}$$

Using any $2t_0$ equations from (4.12) is enough to calculate the two unknown polynomials $\boldsymbol{A}(x)$ and $\boldsymbol{B}(x)$. The problem is that we do not know the set of error positions Ψ. Given the reliability information, we can choose a set υ of $2t_0$ positions from the least reliable positions. Erasures are introduced by forcing a set of positions υ to satisfy the following system of equations:

$$\widehat{\boldsymbol{\Lambda}}(\alpha^i) = \widehat{\boldsymbol{A}}(\alpha^i)\boldsymbol{U}(\alpha^i) + \widehat{\boldsymbol{B}}(\alpha^i)\boldsymbol{V}(\alpha^i) = 0, \quad \text{where } i \in \upsilon. \tag{4.13}$$

If $\upsilon \in \Psi$, then $\widehat{\boldsymbol{A}}(x) = \boldsymbol{A}(x)$ and $\widehat{\boldsymbol{B}}(x) = \boldsymbol{B}(x)$ and $\boldsymbol{\Lambda}(x)$ is calculated using (4.10). However, if $\upsilon \notin \Psi$, we might end up with a polynomial $\widehat{\boldsymbol{\Lambda}}(x)$ instead of $\boldsymbol{\Lambda}(x)$. We would have try again and choose a different set υ.

Chase-like Decoder

Any $2t_o$ error positions should be enough to solve Equation (4.13). If t was known, it is guaranteed to find $2t_0$ error positions in any $n - t + 2t_0$ positions since $n - t$ positions are correct. Then, we obtain possible solutions of $\widehat{\boldsymbol{\Lambda}}(x)$ by substituting in Equation (4.13) for all $\binom{n-t+2t_0}{2t_0}$ possible combinations. By solving the resulting linear system of equations for each combination, we get a list of polynomials. Some having t roots and one of those is guaranteed to be the error locator polynomial $\boldsymbol{\Lambda}(x)$. Unfortunately, this brute force decoding has a large list size due to the huge number of combinations, which results in a high decoding complexity. However, if reliability information is used, we would not need to go through $n - t + 2t_0$ positions. We would only need to choose the least reliable L positions (since they are highly likely to contain more errors than other positions) and search for $2t_0$ error positions in them. Thus, reducing the combinations that should be considered to $\binom{L}{2\tau_0}$ combinations.

Since t is not known at the receiver side, let $\widehat{t} = \tau + \widehat{t_0}$ be the designed decoding radius. The parameters L and $\widehat{t_0}$ are our design parameters. They are limited by how much complexity we can afford. The choice of L should reflect the amount of errors that could be found in the least reliable L positions, such that we are able to calculate the correct $\boldsymbol{\Lambda}(x)$ with low probability of failure. Without loss of generality, assume that the reliabilities for the vector $\boldsymbol{r}$ are sorted in an ascending order such that $\eta_0 \leq \eta_1 \leq \cdots \leq \eta_{n-1}$, such that the chosen positions are the L first positions ($\mathcal{L} = \{0, \ldots, L-1\}$). Since L should be greater than $2\widehat{t_0}$

and is dependent on the quality of the reliability information $\boldsymbol{\eta}$, it is considered as a function of $\widehat{t}_0$. Let $\mathcal{L}^{(\kappa)}$ be the set containing the first $L(\kappa)$ least reliable positions. For the set $\mathcal{L}^{(\kappa)}$ to contain the required number of erroneous positions, $L(\kappa)$ should be designed depending on the quality of the reliability information. However, the larger L becomes, the more time it will take to go through all combinations.

The proposed Chase-like algorithm is an iterative algorithm. In the first iteration, all subsets of size two within the set $\mathcal{L}^{(1)}$ are used to try and solve Equation (4.13) for a $\widehat{\boldsymbol{\Lambda}}(x)$ with degree $\tau+1$. For each successive iteration, the current decoding radius is increased to $\tau+\kappa$ with $\kappa = 1, 2 \ldots, \widehat{t}_0$ until we reach our designed decoding radius $\widehat{t} = \tau + \widehat{t}_0$. Within each iteration, we try subsets of $\mathcal{L}^{(\kappa)}$ of size 2κ in order to get a $\widehat{\boldsymbol{\Lambda}}(x)$ that has a number of roots equal to its degree. This corresponds to $\binom{L(\kappa)}{2\kappa}$ different linear systems of equations for each iteration.

At the end of the algorithm, the output of the decoder is a list of valid solutions for $\boldsymbol{\Lambda}(x)$ having a number of roots equal to their degrees. A success is declared when the list is not empty. Otherwise, a failure is declared. A correct solution occurs when the correct $\boldsymbol{\Lambda}(x)$ is in the list. We consider a wrong solution as an undetected failure. The list size has an upper bound of $\sum_{\kappa=1}^{\widehat{t}_0} \lfloor L(\kappa)/2\kappa \rfloor$, which occurs only if every 2κ in our $L(\kappa)$ least reliable symbols result in a possible solution, which rarely happens as shown later in the numerical simulations. The Chase-like algorithm is shown in Algorithm 4.1.

Single Iteration Chase-like

The algorithm can be made in a single iteration by setting $\kappa = \widehat{t}_0$ directly. Let the difference between the designed radius and the number of errors be $\delta = \widehat{t} - t = \widehat{t}_0 - t_0$. In the case of $t < \widehat{t}$, we have more unknowns in (4.13). The calculated $\widehat{\boldsymbol{\Lambda}}(x)$ will have δ more roots than the correct $\boldsymbol{\Lambda}(x)$. These extra roots are roots at non-erroneous positions which can be removed by just checking whether has also roots at these positions. The polynomial $\widehat{\boldsymbol{\Omega}}(x)$ is calculated along with $\widehat{\boldsymbol{\Lambda}}(x)$ using the EEA and should satisfy the following relations:

$$(x^n - 1) = \boldsymbol{\Lambda}(x)\boldsymbol{\Gamma}(x) = \widehat{\boldsymbol{\Lambda}}(x)\widehat{\boldsymbol{\Gamma}}(x), \tag{4.14}$$

$$\boldsymbol{E}(x) = \boldsymbol{\Omega}(x)\boldsymbol{\Gamma}(x) = \widehat{\boldsymbol{\Omega}}(x) \cdot \widehat{\boldsymbol{\Gamma}}(x). \tag{4.15}$$

Based on Definition 3.1, any root we move from $\boldsymbol{\Gamma}(x)$ to $\boldsymbol{\Lambda}(x)$ is also moved to $\boldsymbol{\Omega}(x)$. The only difference here is that not $2\widehat{t}_0$ erroneous positions are needed

Algorithm 4.1 Chase-like Algorithm

Input: $\boldsymbol{U}(x)\boldsymbol{V}(x), \boldsymbol{\eta}, \widehat{t}_0)$
Initializations: Empty list $\Xi = \{\}$
Set $L(\kappa)$ depending on $\boldsymbol{\eta}$

```
for κ = 1 : t̂_0 do
    L^(κ) ← {0, 1, ..., L(κ) − 1}
    Z ← {z : z ⊂ L^(κ), |z| = 2κ}
    for z ∈ Z do
        Solve ∀i ∈ z: Λ̂(α^i) = A(α^i)U(α^i) + B(α^i)V(α^i) = 0
        if Λ̂(x) is a valid solution then
            Ξ ← Ξ ∪ Λ̂(x)
        end if
    end for
end for
```

Output: A list Ξ of valid possible solutions for $\boldsymbol{\Lambda}(x)$.

to solve (4.13), but $2\widehat{t}_0 - \delta$ erroneous positions and δ non-erroneous positions to find $\widehat{\boldsymbol{\Lambda}}(x)$.

Performance and Complexity

Assume that a position i being erroneous is with probability p_i. Let $P(j)$ denote the probability of j errors occurring in the received vector $\boldsymbol{r}$ and $P(j \notin \mathcal{L})$ denotes the probability that the set $\mathcal{L}$ does not contain j errors or more, which can be calculated by:

$$P(j \notin \mathcal{L}) = \sum_{l=0}^{j-1} \sum_{\binom{L(j)}{l}} \prod_{l} p_i \prod_{L(j)-l} (1 - p_i). \tag{4.16}$$

Then, the probability of a decoding failure P_f, which also includes wrong decoding, can be expressed as follows:

$$P_f = \sum_{j>\widehat{t}} P(j) + \sum_{\kappa=1}^{\widehat{t}_0} P(\tau + \kappa) P(2\kappa \notin \mathcal{L}^{(\kappa)}). \tag{4.17}$$

From (4.16,4.17), it is shown that the probability of failure P_f depends entirely on the quality of the reliability information. The more erroneous positions that are included in the set $\mathcal{L}$, the lower the probability of failure.

The complexity C of the Chase-like algorithm with a designed decoding radius $\widehat{t}$ and probability of failure P_f is mainly based on solving small linear systems of equations with 2κ unknowns for $\binom{L(\kappa)}{2\kappa}$ times for every iteration.

$$\mathrm{C} \sim \sum_{\kappa=1}^{\widehat{t}_0} \binom{L(\kappa)}{2\kappa} (2\kappa)^3. \tag{4.18}$$

Since the complexity is is exponential in $L(\kappa)$, this expression can be approximated to the dominant term:

$$\mathrm{C} \sim \binom{L(\widehat{t}_0)}{2\widehat{t}_0} (2\widehat{t}_0)^3. \tag{4.19}$$

The term $L(\widehat{t}_0)$ is the biggest contributor to the complexity and has to be minimized to achieve low complexity. However, to be able to choose a small $L(\widehat{t}_0)$, the reliability measure needs to maximize the number of errors in the least reliable symbols.

Numerical Evaluations

Using an $\mathcal{RS}(63,57)$ code with BPSK (Binary Phase shift Keying) modulation, transmission over an AWGN channel with SNR varying from 5.2 to 6.4 dB is simulated. At the receiver, the reliability vector $\boldsymbol{\eta}$ is calculated using the method proposed in Section 4.1.2. Based on the calculated vector $\boldsymbol{\eta}$, the probability that the least reliable positions are erroneous is investigated and shown in Figure 4.1. The figure shows that the probability is monotonically decreasing with the positions reliability. With the current setup, the highest probability of error occurs at the least reliable position with a probability varying between 30% and 50%. This means that, on average, less than half of the least reliable position are errors. To achieve correct decoding all the time, $2t_0$ errors should exist in the $L(t_0)$ least reliable positions. Therefore, the value of L should be designed such that $L(\kappa) > 4\kappa$.

The performance of the Chase-like decoder is compared to other decoders that utilize reliability information: A basic GMD decoder and an advanced KV decoder. Note that the GMD decoder is based on the fact that among the L to-be-erased positions there should be more than $L/2$ errors, which is not the case as shown in Figure 4.1. Hence, a bad performance is expected from GMD. Nonetheless, it is used as a reference to see how much better the new decoder performs. The complexity of KV is $\mathcal{O}(\eth^3)$, where $\eth$ is the so-called *cost*. In [MT11], it was proved that decoding with KV can be achieved with

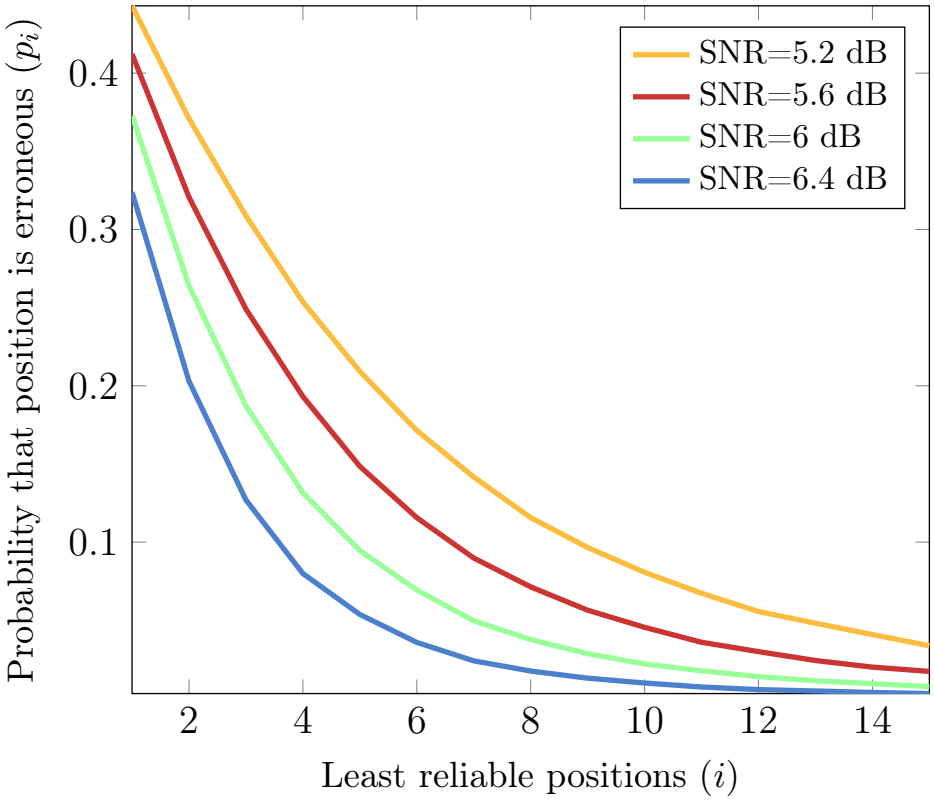

Figure 4.1: Probability of least reliable positions being erroneous. Symbols of codewords from an $\mathcal{RS}(63,57)$ code with BPSK modulation are transmitted over an AWGN channel [MB15].

a lower complexity, namely $\mathcal{O}(\tilde{\delta}^2)$. Up to our knowledge this is lowest known complexity for the KV algorithm. We compare both Chase-like and KV such that the complexities of both algorithms are similar. In order to get simulation results without going through the complex implementation of the KV decoder, we only checked if the bound in [KV03, Theorem 3] is satisfied.

In Figure 4.2, the probability of failing to recover the transmitted codeword (block failure probability) is plotted using different decoders for various SNR values. The Chase-like decoder has a designed decoding radius $\widehat{t} = \tau + 2$. Two variants of the decoders are investigated: one with $L(\kappa) = 4\kappa$ and the other more complex version with $L(\kappa) = 10\kappa$. Also a genie decoder that can correct all $t \leq 5$ errors is simulated as a lower bound. The complexities of both KV and the Chase-like ($L = 10\kappa$) algorithms are equal. From Figure 4.2, it can be seen that

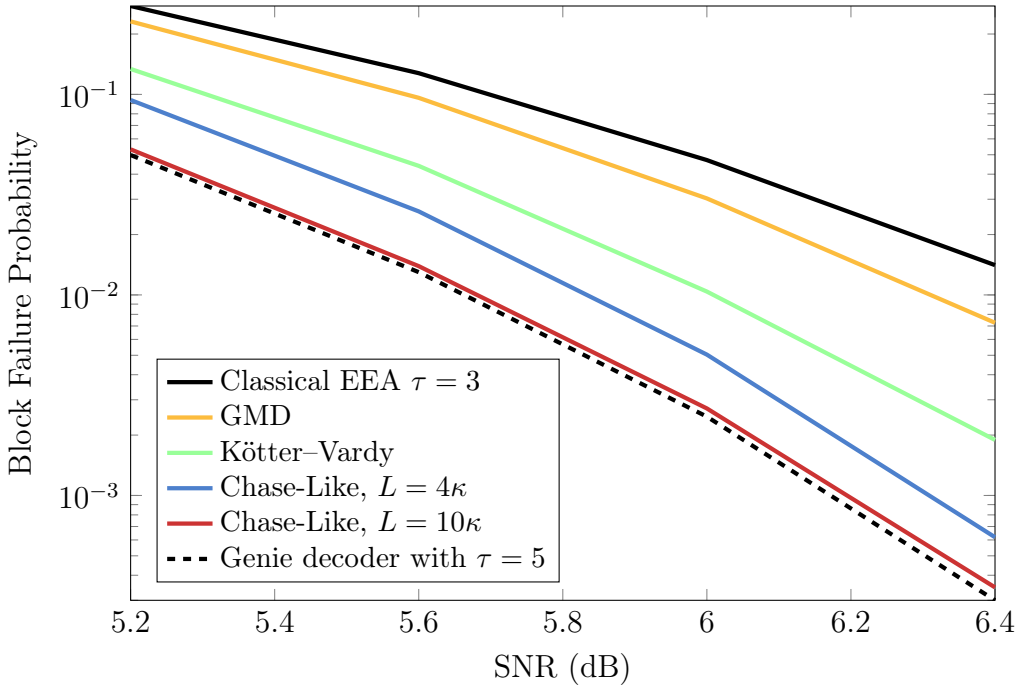

Figure 4.2: Probability of failure vs. SNR. Symbols of codewords from an $\mathcal{RS}(63,57)$ code with BPSK modulation are transmitted over an AWGN channel [MB15].

the Chase-like decoder with both its variants outperforms both the GMD and KV decoders. Although it was only expected for the GMD to be surpassed, the KV lacked the sufficient resources to perform adequately due to the complexity limitation. The more complex version of Chase-like ($L = (10\kappa)$) is almost as good as the genie decoder that can correct extra two errors. Simulations also show that the Chase-like output list size is equal to 1 for $t = 1, 2$.

So in conclusion, if a good enough reliability vector $\boldsymbol{\eta}$ is available, we can correct a small number of errors beyond half the minimum distance τ with a relatively reasonable complexity and a small probability of failure compared to known algorithms when used with high-rate codes. The main disadvantage of the Chase-like decoder is that it has exponential complexity in $L(\tau_0)$, which puts limitations on the designed decoding radius $\widehat{t}$.

4.2.2 Reduced List-Decoder (RLD)

The algorithm in this section was first presented in [MNB14]. The idea of this algorithm is based on the Wu algorithm mentioned in Section 3.2.1. We focus on the case where the decoding of the vector $\boldsymbol{r}$ using EEA (or BMA) is unsuccessful. That is when $t > \tau$, where decoding fails in finding the error locaor polynomial $\boldsymbol{\Lambda}(x)$. In this case, we proceed with rational interpolation while utilizing the outputs obtained from EEA. The goal of rational interpolation in the Wu algorithm is to find a polynomial $\boldsymbol{Q}(x, y, z)$ satisfying the conditions in Theorem 3.4. But instead of using all n points from (3.30) in the interpolation, we only consider to the least reliable positions, obtained from the reliability vector $\boldsymbol{\eta}$. Interpolating with smaller number of points would result in lower complexity. Also a better performance is expected when compared to the original Wu algorithm. This is due to the fact that positions taken into consideration should contain more errors than those that are reliable. Whether the interpolation process succeeds as intended depends on the ratio of errors caught in the unreliable positions. Without loss of generality, assume that the reliabilities for the vector $\boldsymbol{r}$ are sorted in an ascending order such that $\eta_0 \leq \eta_1 \leq \cdots \leq \eta_{n-1}$, such that the chosen positions are the L first positions ($\mathcal{L} = \{0, \ldots, L-1\}$). The points which are used in the interpolation are as follows:

$$\beta_i = -\frac{\boldsymbol{V}(\alpha^i)}{\boldsymbol{U}(\alpha^i)} = \frac{\boldsymbol{A}(\alpha^i)}{\boldsymbol{B}(\alpha^i)}, \quad \forall i = 0, \ldots, L-1. \tag{4.20}$$

The coprime polynomials $\boldsymbol{U}(x)$ and $\boldsymbol{V}(x)$ are outputs of the EEA (or BMA) at a certain iteration, while $\boldsymbol{A}(x)$ and $\boldsymbol{B}(x)$ are unknown polynomials satisfying

$$\begin{aligned} \deg \boldsymbol{A}(x) &= \theta_A = \deg \boldsymbol{\Lambda}(x) - \deg \boldsymbol{U}(x) \\ \deg \boldsymbol{B}(x) &= \theta_B \leq \deg \boldsymbol{\Lambda}(x) - d - \deg \boldsymbol{U}(x) \end{aligned} \tag{4.21}$$

Since t stands for the total number of errors, let t_L be the number of errors in the L chosen positions. Because both of them are unknown to the receiver, we assume the two values: $\widehat{t}$ and $\widehat{t}_L \leq \widehat{t}$. Choosing $\widehat{t}$ depends on the maximum number of errors one wishes to correct. As for $\widehat{t}_L$, one would assume that it should be chosen such that it somehow relates to t_L. However, it is not that straightforward and we will get back to choosing this value later on. A visualisation of the parameters is shown in Figure 4.3.

Provided with Theorem 3.3, we then formulate a rational interpolation problem for finding $\boldsymbol{A}(x)$ and $\boldsymbol{B}(x)$ using the obtained polynomials $\boldsymbol{U}(x)$ and $\boldsymbol{V}(x)$:

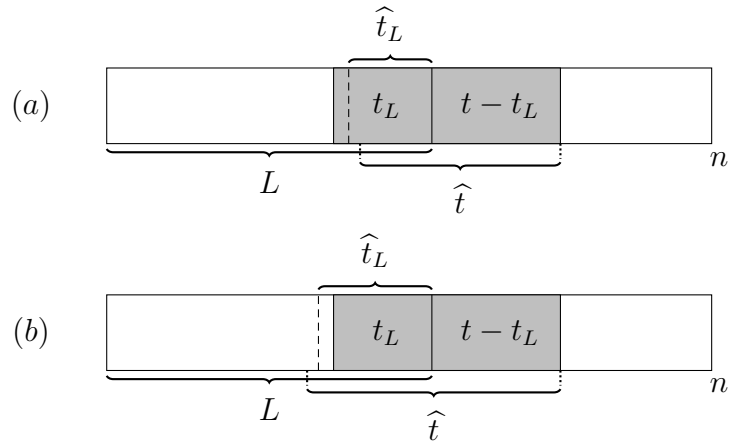

Figure 4.3: Visualisation of L, t, t_L, $\widehat{t}$ and $\widehat{t}_L$. Error positions have been ordered for overview. In (a) $t > \widehat{t}$ and in (b) $t < \widehat{t}$, but in either case decoding succeeds since t_L is sufficiently large, as according to Lemma 4.2.

Lemma 4.1
Consider Theorem 3.3 and let the number of interpolation points $N = L$ and $(x_i, y_i : z_i) = \big(\alpha_i, \boldsymbol{A}(\alpha^i) : \boldsymbol{B}(\alpha^i)\big)$ for $i = 0, \ldots, L-1$, as well as $T = \widehat{t}_L$, $\theta_A = \widehat{t} - \deg \boldsymbol{A}(x)$ and $\theta_B = \widehat{t} - d + \deg \boldsymbol{B}(x)$. If

$$\widehat{t}_L^2 > L(2\widehat{t} - d), \tag{4.22}$$

there exist valid choices of s and ℓ such that a polynomial $\boldsymbol{Q}(x, y, z)$ satisfying the requirements of Theorem 3.3 exists. Furthermore, if $t = \widehat{t}$ and $t_L = \widehat{t}_L$, then $\boldsymbol{Q}(x, \boldsymbol{A}(x), \boldsymbol{B}(x)) = 0$.

Proof. The existence of the $\boldsymbol{Q}$-polynomial follows directly from Theorem 3.3 and (3.32). The property $\boldsymbol{Q}(x, \boldsymbol{A}(x), \boldsymbol{B}(x)) = 0$ follows from the arguments of Section 3.2.1 since the equation $\boldsymbol{A}(\alpha^i)\boldsymbol{U}(\alpha^i) + \boldsymbol{B}(\alpha^i)\boldsymbol{V}(\alpha^i) = 0$ holds for $T = \widehat{t}_L$ out of $N = L$ values of i, and since $\deg \boldsymbol{A}(x) + \deg \boldsymbol{B}(x) \leq 2t - (\deg \boldsymbol{U}(x) + \deg \boldsymbol{V}(x)) = 2\widehat{t} - d = \theta_A + \theta_B$. □

Lemma 4.1 shows that the desired $\boldsymbol{Q}$-polynomial is obtainable only for the case where the chosen value $\widehat{t}$ is equal to the number of errors t. Nevertheless, we have to also consider if that is also possible whenever $\widehat{t} \neq t$. For the original Wu algorithm, it is proven that success is guaranteed even for $\widehat{t} > t$. Here the situation turned out to be surprisingly different.

Lemma 4.2
Considering Lemma 4.1 when $t \neq \widehat{t}$*, then* $\boldsymbol{Q}(x, \boldsymbol{A}(x), \boldsymbol{B}(x)) = 0$ *if:*

$$\frac{\ell}{s} \geq \frac{\widehat{t}_L - t_L}{\widehat{t} - t}, \quad \textit{whenever } \widehat{t} > t, \tag{4.23}$$

$$\frac{\ell}{s} \leq \frac{\widehat{t}_L - t_L}{t - \widehat{t}}, \quad \textit{whenever } \widehat{t} < t. \tag{4.24}$$

Proof. This statement is proven by showing that $\boldsymbol{Q}$ is a valid interpolation polynomial satisfying the requirements of Theorem 3.3 for almost the same rational interpolation problem but with $\widetilde{T} = t_L$ and $\widetilde{\theta}_A = t - \deg \boldsymbol{U}(x)$ and $\widetilde{\theta}_A = t + d - \deg V(x)$.

Since only T, θ_A and θ_B are changed in the newly considered rational interpolation, we only need to show that the new weighted-degree constraints on $\boldsymbol{Q}$ are satisfied, i.e. that:

$$\deg_{(1,\widetilde{\theta}_A,\widetilde{\theta}_B)} \boldsymbol{Q} < st_L. \tag{4.25}$$

Since $\boldsymbol{Q}$ satisfied the original interpolation problem, then $\deg_{(1,\theta_A,\theta_B)} \boldsymbol{Q} < s\widehat{t}_L$. We then compute

$$\begin{aligned} \deg_{(1,\widetilde{\theta}_A,\widetilde{\theta}_B)} \boldsymbol{Q} &\leq \deg_{(1,\theta_A,\theta_B)} \boldsymbol{Q} && (4.26)\\ &\quad - \min_{i=0,\ldots,\ell} \left\{ i(\theta_A - \widetilde{\theta}_A) + (\ell - i)(\theta_B - \widetilde{\theta}_B) \right\} && (4.27)\\ &= \deg_{(1,\theta_A,\theta_B)} \boldsymbol{Q} - \ell(\widehat{t} - t). && (4.28) \end{aligned}$$

Thus $\deg_{(1,\theta_A,\theta_B)} \boldsymbol{Q}$ is satisfactory low whenever

$$st_L \geq s\widehat{t}_L - \ell(\widehat{t} - t),$$

which is equivalent to the conditions of the lemma. □

From Lemma 4.2, it turns out that, similar to the Wu algorithm, rational interpolation succeeds even for the case of $\widehat{t} > t$. This can be even done when fewer errors than $\widehat{t}_L$ exist in the L chosen positions. Surprisingly, however, it turns out that even when $\widehat{t} < t$, rational interpolation succeeds if more errors exist in the chosen positions. As a result, it would be a mistake to call $\widehat{t}$ a decoding radius, since success is guaranteed (under some conditions) even when errors beyond this value occur. Instead of radius, we would call $\widehat{t}$ the decoding *pivot*. Equations (4.23) and (4.24), can also be written in the following form:

$$\widehat{t}_L \leq t_L + \frac{\ell}{s}(t - \widehat{t}) \tag{4.29}$$

Algorithm 4.2 shows the complete proposed decoding algorithm. While Theorem 4.1 defines which codewords are contained in the list returned by Algorithm 4.2:

Theorem 4.1
Let $\boldsymbol{r}$,$\boldsymbol{\eta}$ be the received word and its reliability vector respectively. Let $L, \ell, s, \widehat{t}, \widehat{t}_L$ be parameters as defined in Algorithm 4.2. If $\exists\ \boldsymbol{c} \in \mathcal{RS}$ s.t. $\mathrm{wt_H}(\boldsymbol{c}-\boldsymbol{r}) \leq \tau$, then $\boldsymbol{c}$ is returned. Otherwise, the set $T \subset \mathcal{RS}$ is returned s.t $\boldsymbol{c} \in T$ iff:

$$\mathrm{wt_H}(\boldsymbol{c}-\boldsymbol{r})_L \geq \widehat{t}_L - \frac{\ell}{s}(\widehat{t} - \mathrm{wt_H}(\boldsymbol{c}-\boldsymbol{r})), \tag{4.30}$$

where $\mathrm{wt_H}(\boldsymbol{x})_L$ is the Hamming weight of $\boldsymbol{x}$ within the least reliable L positions according to $\boldsymbol{\eta}$.

Proof. Follows from Lemma 4.1 and Lemma 4.2. □

Algorithm 4.2 RS Reduced list-decoding with reliability information

Require: A code $\mathcal{RS}(n,k)$ over $\mathbb{F}_q$ as in Definition 2.9.
The vector $\boldsymbol{r} \in \mathbb{F}_q^n$ with its corresponding reliability vector $\boldsymbol{\eta}$.
The decoding pivot $\widehat{t}$ and the number of interpolation points L.

Ensure: A list of codewords in $\mathcal{RS}$ or Fail.

1: Calculate the syndrome $\boldsymbol{S}(x)$ from $\boldsymbol{r}$ as in (3.4).
2: Run the EEA on $x^{d-1}, \boldsymbol{S}(x)$ and calculate $\boldsymbol{U}(x), \boldsymbol{V}(x)$.
3: If $\boldsymbol{U}(x)$ is a valid error-locator of degree less than $d/2$, use it to correct $\boldsymbol{r}$, and if this yields a word in $\mathcal{RS}$, return this one word.
4: Otherwise, we seek $\boldsymbol{A}(x), \boldsymbol{B}(x)$ as in (4.10). Set $\widehat{t}_L = \left\lfloor \sqrt{L(2\widehat{t}-d)} + 1 \right\rfloor$ and set θ_A, θ_B as in Lemma 4.1. Construct the $\boldsymbol{Q}(x,y,z)$ described in that lemma, using satisfactory values of s and ℓ.
5: Find all pairs of polynomials $\boldsymbol{A}^\star(x)$ and $\boldsymbol{B}^\star(x)$ such that $\boldsymbol{Q}(x, \boldsymbol{A}^\star(x), \boldsymbol{B}^\star(x)) = 0$. Return Fail if no such pairs exist.
6: For each such pair, construct $\boldsymbol{\Lambda}^\star(x) = \boldsymbol{A}^\star(x)\boldsymbol{U}(x) + \boldsymbol{B}^\star(x)\boldsymbol{V}(x)$. If it is a valid error-locator, use it for correcting $\boldsymbol{r}$. Return Fail if none of the factors yield error-locators.
7: Return those of the corrected words that are in $\mathcal{RS}$. Return Fail if there are no such words.

Classical minimum distance decoding takes place until Step 3 in Algorithm 4.2, which can be performed with a complexity of $\mathcal{O}(n \log^{2+o(1)} n)$. The rest of the algorithm is exactly as the steps of the Wu algorithm. However, in our variant, we only use L of the total n points. This reduces the complexity of the list decoding part to a complexity of $\mathcal{O}(\ell^M sL \log^{\mathcal{O}(1)}(\ell L))$.

Numerical Evaluation

Before we go to the actual results, we discuss first how the parameters of the RLD are chosen. How to optimally choose the parameters $L, \widehat{t}, \widehat{t}_L, \ell$ and s used in Algorithm 4.2 is not yet clear. For example, there is no actual decoding radius; rather a decoding pivot $\widehat{t}$. It should not be looked upon as the maximum number of the errors that can be corrected, since the decoder manages to correct even beyond it as shown in Lemma 4.2. Still, for a specific case, we later show that the approach used to choose these values is fairly plausible.

From (4.22), it is clear that the success of the interpolation step will depend on both L and τ_L. Also from (4.29), the goal would be to catch as many t_L errors as possible in the least L reliable positions. Provided we have good reliability information, we could lower the complexity of the decoder and choose a small value for L since most of the positions would be erroneous. While in the case of bad reliability information, one would need to consider more points to achieve an adequate performance; at the cost of complexity of course. In conclusion, the value of L should be dependant on the quality of the reliability information provided by $\boldsymbol{\eta}$. Since there are no restrictions on L, we simulate using small, medium and large values of L to see the effect of the change. From Lemma (4.2), we know that the value of $\widehat{t}_L$ does not necessarily need to be equal to t_L. Actually, (4.29) shows that $\widehat{t}_L$ should be chosen as minimum as possible while satisfying (4.22). Thus, the perfect choice for τ_L would be $\tau_L = \lfloor \sqrt{L(2\tau - d)} + 1 \rfloor$.

To calculate the list size ℓ and multiplicity s, we use the method proposed by Trifonov in [Tri10]. This method produces the numerically smallest possible values for ℓ and s, which would certainly minimize the computational complexity of the simulations. However, one must not ignore the fact that according to Lemma 4.2 the ratio ℓ/s has an impact on the decoding performance. For example, if $\widehat{t} > t$, the ratio ℓ/s should be chosen as large as possible to maximize the probability of success. While in the case that $\widehat{t} < t$, it should be as small as possible. This change will definitely result in a change of the complexity; an increase to more be precise. In any case, the number of errors t is not known, so there is not much that can be done. A suggestion would be to spilt every decoding trial into two trials with each having different ratios ℓ/s; one maximized

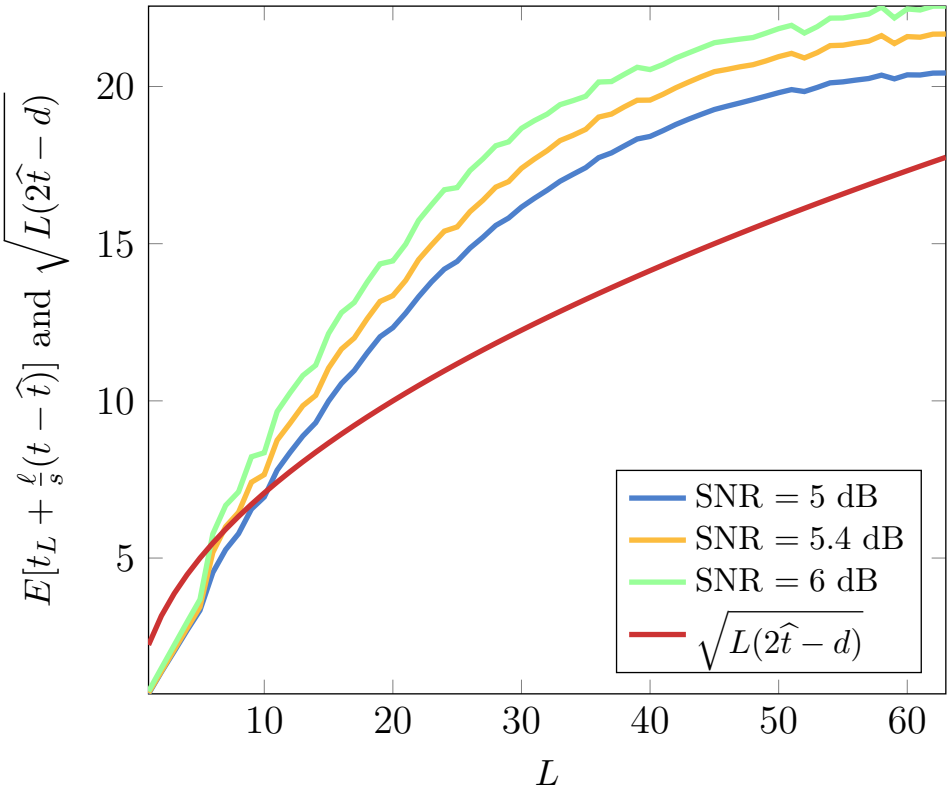

Figure 4.4: $E[t_L + \frac{\ell}{s}(t - \widehat{t})]$ for L least reliable positions vs $\sqrt{L(2\widehat{t} - d)}$. Symbols of codewords from an $\mathcal{RS}(63,31)$ code with BPSK modulation are transmitted over an AWGN channel.

and the other minimized. Another suggestion would be to simply increase the decoding pivot $\widehat{t}$, such that we are more likely to fall in the case where $\widehat{t} > t$. These suggestions are yet to be proved useful and should be subject to future investigation. Also the optimized values for the parameters at hand are yet to be established. This means that there is still room for possible improvement in the algorithms performance.

We compare our RLD with two other decoding algorithms. The Wu algorithm is chosen as our hard-decision decoder to see the improvement when using reliability information. The second decoder is chosen as a rival that also uses reliability information. For that purpose, we decided to chose one of the best performing decoders, the KV algorithm. Symbols of codewords from the code

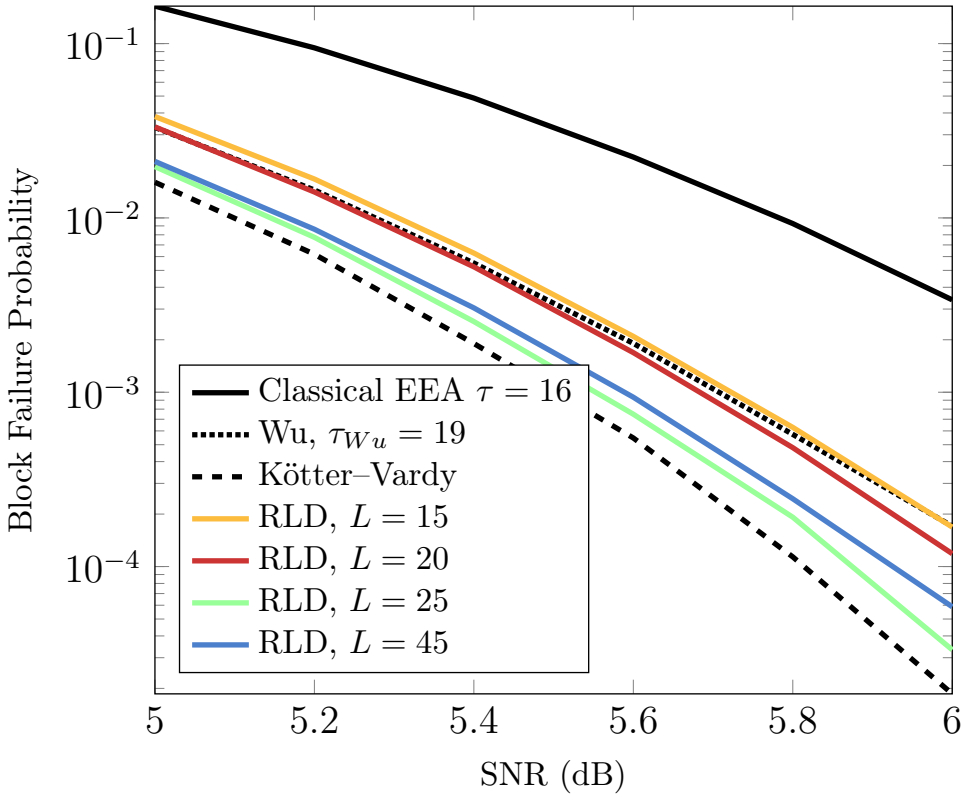

Figure 4.5: Probability of failure vs. SNR. Symbols of codewords from an $\mathcal{RS}(63,31)$ code with BPSK modulation are transmitted over an AWGN channel.

$\mathcal{RS}(63, 31)$ with BPSK modulation are transmitted over an AWGN channel while varying the SNR. The reliability vector $\boldsymbol{\eta}$ is calculated as described in Section 4.1.2. For each SNR, the transmission of a total number of one million codewords is simulated. For the $\mathcal{RS}(63, 31)$ code, the half minimum decoding radius is $\tau = 16$, while the maximum possible hard-decision list-decoding radius of the Wu decoder is $\tau_{Wu} = 19$. We chose the decoding pivot $\widehat{t} = 19$, similar to that of the Wu algorithm, to see the improvement of using reliability information provided by the vector $\boldsymbol{\eta}$.

As mentioned before, if (4.29) is satisfied, decoding succeeds. To get a feeling on the effect of choosing different values of L and consequently $\widehat{t}_L$, the right-hand-side of (4.29) is investigated. Our simulation consist of a million $\mathcal{RS}(63, 31)$ codewords such that codeword symbols are transmitted with BPSK

modulation over an AWGN channel. The value $E[t_L + \frac{\ell}{s}(t - \widehat{t})]$ is calculated at various SNRs, where $E[x]$ denotes the expected value of x.

In Figure 4.4, the red curve corresponds to $\sqrt{L(2\widehat{t} - d)}$ which is the lower bound for $\widehat{t}_L$ allowing rational interpolation to succeed (see Lemma 4.1). The rest of the curves show the expected value of $t_L + \frac{\ell}{s}(\widehat{t} - t)$ at different SNRs. For a million codewords, it was expected that the curves obtained would turn out smoother than what is shown in Figure 4.4. This would be the case if it were not for the effect of L on the choices of ℓ and s. For each value of L, we get a different ratio ℓ/s, which sometimes suddenly increase or decrease; thus explaining the rugged behaviour of the curves. Based on this figure, we can decide what values for L that corresponds in a better value for $\widehat{t}_L$. For example, choosing an $L < 10$ would result in a $\widehat{t}_L$ which is way larger than the value $E[t_L + \frac{\ell}{s}(t - \widehat{t})]$ resulting in a decoding failure. So for this specific code and SNRs, a number of points $L > 10$ should be used in the interpolation. With the increase of SNR, we see the gap between the bound and the other curves increase. This indicates that a better success rate would result with a higher SNR. A final conclusion from Figure 4.4 would be that having an L that results in a large gap between the bound and the other curves would result in a better performance, which turns out to be true.

In Figure 4.5 the probability of a decoding failure is plotted for the chosen code while varying the SNR from 5 to 6 dB. An RLD decoding failure in this simulation is either the decoder producing an empty list or a list that does not contain the transmitted codeword.

Up to our knowledge, the KV algorithm is considered one of the best reliability information decoders. This however comes at the cost of the decoding complexity. For the purpose of this simulation, we assumed that the KV decoder has a really high complexity. This is achieved by setting its maximum multiplicity-sum to the value of $2n$, which can be considered, in terms of complexity, an expensive decoder. As expected, it outperforms all other decoders as seen in Figure 4.5.

The figure also shows that using RLD with choosing $L = 15$ and $L = 20$ is almost equivalent to using the Wu algorithm. Note that, the Wu algorithm can be viewed as RLD with $L = n$. Increasing the number of interpolation points to $L = 25$ increases the performance of RLD drastically. Resulting in a performance that is almost equal to that of the highly complex KV. Increasing the L further to 45 degrades the performance. If we increase L even further to 63, we get the performance of the Wu algorithm. This is expected since the curves from Figure 4.4 start getting closer to the lower bound, thus satisfying

(4.29) less often.

It is worth mentioning, that the value of the ration ℓ/s is $5/3$ for $L = 15, 20$, 2 for $L = 25$, and 3 for $L = 45$. According to (4.29), these values play a important role in the success rate of the decoder and, hence, should be subjected to further investigation.

4.3 Overview and Summary

This chapter focused on algebraic decoding of RS codes over finite fields with the help of reliability information. Using already established concepts and algorithm like GMD, Chase, Wu and Kötter–Vardy, we show the performance of new methods. The goal was to achieve either a better performance from that of the known algorithms or the same performance using much less computational complexity.

First, the Chase-like decoder [MB15] is investigated. It is shown that correcting one or two extra errors using this decoder does not add much of computational complexity to the basic EEA (BMA) decoder. However, correcting even more errors would increase the complexity of the decoder exponentially. A single iteration Chase-like decoder presents a chance of a slight speed up but at the cost of performance. This is due to the oversight of combinations that would result in a decoding success. Compared to KV decoder with limited computational power, the Chase-like decoder is able to outperform it. Still, this good performance hinges on the existence of good reliability information, since one is required to catch a certain number erroneous positions in the least reliable ones.

On the other hand, when an interpolation-based decoder like Wu is used, such a strict condition does not exist. Provided with reliability information, a reduced version of Wu was established [MNB14]. The RLD algorithm, similar to the Wu algorithm, uses rational interpolation as a tool for decoding. The difference is that the RLD does not use all the points provided by the received vector and uses only those that are likely to be erroneous. In terms of complexity, the new decoder offers less computational complexity than that of the Wu algorithm. Simulations show that RLD greatly exceeds the Wu algorithm in performance and is comparable with an extremely expensive KV decoder. Even so, the RLD did not yet reach its maximum potential. There are still parts that need to be investigated regarding the choice of parameters. It is clear that with the proper optimization, the RLD would produce even better results.

5

Decoding with Reliability in the Complex Field

Complex Reed–Solomon codes (CRS) are first presented by Wolf [Wol83] and Marshall [Mar84]. They are defined over the complex field $\mathbb{C}$ using DFT, so they were referred to at the time as analog or DFT codes. Even though the digital revolution has already changed the way computers and communication systems were designed, CRS codes, being defined over $\mathbb{C}$, proved to be an interesting topic for researchers. Some focused on studying and enhancing already existing decoders to suit these type of codes. In [Wol83], the focus was the removal of impulse noise from discrete-time continuous-amplitude data sequence whose DFT has a string of continuous zeros. Having continuous zeros in a data sequence can occur if a continuous band-limited is sampled with a sampling rate higher than the Nyquist rate. While in [Mar84], it is shown that it is possible, under some condition, to correct $d/2$ errors if the distance of the code is even. Extension of CRS codes to achieve a better performance when decoding using BMA and EEA is investigated in [MS85]. In [Kum85], the use of CRS codes to correct burst errors is examined and it is shown that with interleaving better decoding results are achieved.

Other researchers focused on applications for such codes. Examples of these applications are protection against packet loss in IP networks [RHG01] and estimation of the direction of arrival of plane waves [RG03]. They can be also utilized in Orthogonal Frequency-Division Multiplexing (OFDM) [Hen00, HH05, ADAA08, HHH12]. There are also other publications that are worth mentioning such as [Hen89, MHET99, Red00, TH08, AT08, HHZ11, VL14]. However, our motivation for studying CRS codes lies in their application in Compressed Sensing (CS), where reconstruction of sparse vectors from their compressed counterparts is investigated. The utilization of CRS codes in deterministic CS started

by Parvaresh and Haassibi in [PH08], where they used the Coppersmith–Sudan decoding algorithm [CS03]. Motivated by these results, [MRZB15] shows that known decoders such as BMA and EEA can achieve an exceptional performance even when dealing with a noisy scenario of data compression. As a result to this work, it is later established in [Zö15, Chapter 7] that one can extract reliability information whenever the algorithm used fails to recover the sparse vector. This reliability information can be used in another second attempt of sparse vector reconstruction.

In this chapter, we define the CRS codes and how they can be used for CS sparse reconstruction. Then, we show the origin of the built-in reliability information and investigate different methods of reliability information extraction. Finally, we discuss two new decoding algorithms that make use of this reliability information and show their performance.

5.1 Reed–Solomon Codes over the Complex Field

Before redefining RS codes over the complex field, we need first to also redefine the DFT, which is already defined in Definition 2.3 for polynomials over the finite field. From this point onward, we use the following definition for DFT for complex-valued polynomials.

Definition 5.1 (DFT for complex-valued polynomials)
For a given polynomial $\boldsymbol{A}(x) \in {}^{\mathbb{C}[x]}/_{(x^n-1)}$*, the DFT* $\mathcal{F}[\boldsymbol{A}(x)] = \boldsymbol{a}(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}$ *is defined as*

$$a_j = \frac{1}{\sqrt{n}}\boldsymbol{A}(\alpha^j), \quad j = 0, \ldots, n-1, \tag{5.1}$$

and the Inverse DFT (IDFT) $\mathcal{F}^{-1}[\boldsymbol{a}(x)] = \boldsymbol{A}(x)$ *as:*

$$A_j = \frac{1}{\sqrt{n}}\boldsymbol{a}(\alpha^{-j}), \quad j = 0, \ldots, n-1, \tag{5.2}$$

with $\alpha = \mathrm{e}^{-\mathrm{j}\frac{2\pi}{n}}$.

The scalar factor $1/\sqrt{n}$ is the same as that in Definition 2.3, which was only in the IDFT side with the value of $1/n$. Putting this factor on both transform and inverse transform provides some kind of symmetry between the transforms which simplifies future relations.

RS codes over the complex fields are also called Complex Reed–Solomon (CRS) codes. Similar to Definition 2.9, while taking the changes done in the definition of the DFT into consideration, we define CRS codes as follows

Definition 5.2 (Complex Reed–Solomon codes)
Let n and k be positive integers fulfilling $k < n$. The RS code $\mathcal{CRS}(n,k)$ of length n and dimension k is the set

$$\left\{ \frac{1}{\sqrt{n}} \left(\boldsymbol{C}\left(\alpha^0\right), \ldots, \boldsymbol{C}\left(\alpha^{n-1}\right) \right) \middle| \, \boldsymbol{C}(x) \in \mathbb{C}[x] \wedge \deg \boldsymbol{C}(x) < k \right\},$$

where $\alpha = \mathrm{e}^{-\mathrm{j}\frac{2\pi}{n}}$, and $\mathrm{j} = \sqrt{-1}$, such that $\alpha^n = 1$ and $\alpha^i \neq 1$ for $0 < i < n$.

The minimum Hamming distance of the code is $d = n - k + 1$. The corresponding generator matrix $\boldsymbol{G}_{\mathrm{CRS}}$ has the following form:

$$\boldsymbol{G}_{\mathrm{CRS}} = \frac{1}{\sqrt{n}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha^1 & \alpha^2 & \cdots & \alpha^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^{k-1} & \alpha^{2(k-1)} & \cdots & \alpha^{(n-1)(k-1)} \end{pmatrix}, \tag{5.3}$$

and the parity check matrix $\boldsymbol{H}_{\mathrm{CRS}}$:

$$\boldsymbol{H}_{\mathrm{CRS}} = \frac{1}{\sqrt{n}} \begin{pmatrix} 1 & \alpha^k & \alpha^{2k} & \cdots & \alpha^{(n-1)k} \\ 1 & \alpha^{k+1} & \alpha^{2(k+1)} & \cdots & \alpha^{(n-1)(k+1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^{n-1} & \alpha^{2(n-1)} & \cdots & \alpha^{(n-1)(n-1)} \end{pmatrix}. \tag{5.4}$$

For simplicity, we redefine $\boldsymbol{H}$ for this chapter such that $\boldsymbol{H} = \boldsymbol{H}_{\mathrm{CRS}}$. The relation between the parity-check matrix $\boldsymbol{H}$ and a codeword $\boldsymbol{c} \in \mathcal{CRS}$ remains unchanged and is $\boldsymbol{c}\boldsymbol{H}^\dagger = \boldsymbol{0}$, where $\boldsymbol{c}^\dagger$ is the Hermitian conjugate of $\boldsymbol{c}$. A received vector $\boldsymbol{r} \in \mathbb{C}^n$ is the addition of a sparse error vector $\boldsymbol{e} \in \mathbb{C}^n$ to a codeword $\boldsymbol{c}$ such that $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$. A syndrome $\boldsymbol{S}$ is defined for CRS codes as it is defined for RS codes in Equation (3.2), where $\boldsymbol{S} = \boldsymbol{r}\boldsymbol{H}^\dagger = (\boldsymbol{c}+\boldsymbol{e})\boldsymbol{H}^\dagger = \boldsymbol{e}\boldsymbol{H}^\dagger$. Since we are dealing with complex numbers, there is a new factor that appears in the received vector $\boldsymbol{r} = \boldsymbol{c}+\boldsymbol{e}+\boldsymbol{\zeta}$, where $\boldsymbol{\zeta} \in \mathbb{C}^n$ is the noise vector. This noise term represents numerical inaccuracies such as quantization errors, measurement noise and the finite precision of the computation. As a result also the syndrome has its own noise term such that $\boldsymbol{S} = \boldsymbol{e}\boldsymbol{H}^\dagger + \boldsymbol{\zeta}\boldsymbol{H}^\dagger = \boldsymbol{e}\boldsymbol{H}^\dagger + \boldsymbol{\zeta}_S$. The values of the vectors $\boldsymbol{\zeta}$ and $\boldsymbol{\zeta}_S$ are drawn from a random distribution. The type of this distribution and its parameters depend on the application where CRS codes are used and the implementation method of the decoding system and their precision.

Using the syndrome, Wolf showed in [Wol83] that it is possible to correct up to $t \leq d - 2$ errors. Although, this method is exponential in complexity, there

exists no other known algorithm that can achieve such radius. Conventional syndrome-decoding techniques presented in Section 3.1 are able to decode up to half the minimum distance $\tau = \lfloor (d-1)/2 \rfloor$. Even in the presence of some noise, it is possible to achieve successful decoding if the decoders were to be tweaked to operate with CRS codes. This tweaking of algorithms such as the BMA and EEA is shown in [MRZB15]. Further observations on the difference in performance of syndrome-decoding algorithms can be found in [Zö15, Chapter 7]. When referring to one of the algorithms in this chapter, we refer to their CRS-adapted version presented in [MRZB15] and [Zö15, Chapter 7].

5.2 Relation to Compressed Sensing

Compressed Sensing (CS) is becoming more popular over the last few years. It can be claimed that the field gained the attention after [Don06, CT06] were published. It is also worth mentioning that the term "CS" appeared first in [Don06]. Since then, CS was used in many applications [EK12, FR13]. The main objective of CS is to solve an under-determined linear system of equations while searching for the solution with the smallest support; hence, the sparsest solution. The linear system of equations is denoted as

$$\boldsymbol{z}\boldsymbol{D}^{\dagger} = \boldsymbol{b}, \tag{5.5}$$

where $\boldsymbol{D} \in \mathbb{C}^{m \times n}$ is called the sensing matrix, $\boldsymbol{b} \in \mathbb{C}^{m}$ is the measurement vector. For a given $\boldsymbol{D}$ and $\boldsymbol{b}$, calculating the vector $\boldsymbol{z} \in \mathbb{C}^{n}$ is the goal of CS. The process is referred to as reconstruction or recovery. Since it is an under-determined linear system of equations, there are infinitely many solutions. Calling a vector sparse means that it contains a small number of non-zero elements; a vector with a small support. By assuming the vector $\boldsymbol{z}$ to be sparse, obtaining a unique solution becomes possible if it is indeed sparse enough. Searching for the sparsest possible solution for $\boldsymbol{z}$ is considered NP-hard [Nat95]. However, there exist many suboptimal reconstruction algorithms such as the Orthogonal Matching Pursuit (OMP) algorithm [CBL89].

The sensing matrix $\boldsymbol{D}$ is usually chosen to be a random sensing matrix. In order to give bounds on the allowed sparsity for a successful reconstruction of the sparse vector $\boldsymbol{z}$, several properties for the sensing matrix have been proposed. The maximal coherence between the columns of $\boldsymbol{D}$ is a very popular property [DE03]. The matrices presented in [ZB15] where described by Best Complex Antipodal Spherical Codes (BCASCs). To the best of our knowledge, these matrices are the best known matrices to be used for sparse reconstruction

with respect to the coherence and are used for performance comparisons later in the chapter.

CS is called deterministic CS if the matrix $\boldsymbol{D}$ is chosen with a specific non-random structure. For deterministic CS, sensing matrices are constructed in order to have advantageous properties and provide non-probabilistic reconstruction guarantees as well as faster less complex reconstruction [DeV07, AMM12]. Depending on the chosen structure, dedicated algorithms can even offer improved performance compared to general algorithms [CHJ10].

In [PH08], a deterministic CS reconstruction algorithm based on CRS codes is proposed, where reconstruction is done using the Coppersmith–Sudan algorithm [CS03]. In this approach, the matrix $\boldsymbol{D}$ is chosen to be the CRS parity check matrix $\boldsymbol{H}$ as defined in (5.4) with $m = n-k$. If the sparse vector $\boldsymbol{z}$ we are trying to reconstruct is considered as the error vector $\boldsymbol{e}$, the measurement vector $\boldsymbol{b}$ would be simply the syndrome $\boldsymbol{S}$. As a result, Equation (5.5) is transformed to Equation (3.2).

$$\boldsymbol{z}\boldsymbol{D}^{\dagger} = \boldsymbol{b} \quad \Longleftrightarrow \quad \boldsymbol{e}\boldsymbol{H}^{\dagger} = \boldsymbol{S}. \tag{5.6}$$

Equation (5.6) implies that sparse reconstruction in CRS-based CS is equivalent to the decoding of CRS codes. This also indicates that the methods presented in Chapter 3 can be used to recover the sparse vector $\boldsymbol{e}$. The difference between (3.2) and (5.6) is that the $\boldsymbol{e}$, $\boldsymbol{H}$ and $\boldsymbol{S}$ are complex-valued; such that $\boldsymbol{e} \in \mathbb{C}^n$, $\boldsymbol{H} \in \mathbb{C}^{(n-k)\times n}$ and $\boldsymbol{S} \in \mathbb{C}^{n-k}$. Unlike in Chapter 4, our focus is on the application of CRS codes in deterministic CS. Thus, the process of encoding a message and transmitting a codeword does not exist any more. Instead, we have the compression process expressed by (5.6). Syndrome-based decoding methods (see Section 3.1) can be used directly for sparse reconstruction by utilizing the measurement vector $\boldsymbol{S}$ [MRZB15]. On the other hand, to be able to utilize interpolation-based methods (see Section 3.2), the vector $\boldsymbol{r}$ (no longer called a received vector in this chapter) needs to be calculated as follows

$$\boldsymbol{r} = \boldsymbol{H}^{\dagger}\boldsymbol{S} = \boldsymbol{c} + \boldsymbol{e}, \tag{5.7}$$

where $\boldsymbol{c} \in \mathcal{CRS}$ is an arbitrary codeword. The actual value of codeword $\boldsymbol{c}$ does not affect the reconstruction process. Still, in some interpolation-based decoding methods, the calculation of $\boldsymbol{c}$ is required. An example is the GS-based decoder from [MPB17], which is later presented in Section 5.4.2.

Based on the results established in [MRZB15], Zörlein showed in [Zö15, Chapter 7] that when reconstruction with classical algorithms (like BMA) fails due to a large number of errors t (sparsity), one obtains reliability information that

can be used in another advanced step to attempt the reconstruction. This reliability information comes from the fact that syndrome-based decoding methods are considered a Padé approximation [BGM96]. In [Zö15, Algorithm 7.1], Zörlein also presented the Continuity Assisted Decoding (CAD) algorithm which makes use of the reliability information provided by the first attempt of decoding. We later use CAD as reference when checking the performance of the proposed algorithms.

5.3 Padé Approximation-based Reliability

The Padé approximation was first introduced by Henry Padé in [Pad92]. This mathematical problem can be seen in many fields such as coding theory, convergence theory and quantum mechanics [BGM96]. For a given function $\boldsymbol{f}(x)$ with the power series

$$\boldsymbol{f}(x) = \sum_{i=0}^{\infty} f_i x^i, \tag{5.8}$$

the Padé approximation $[\mu/\nu]_{\boldsymbol{f}}(x)$ is the rational function

$$[\mu/\nu]_{\boldsymbol{f}}(x) = \frac{\boldsymbol{a}(x)}{\boldsymbol{b}(x)} = \frac{a_0 + a_1 x + \cdots + a_\mu x^\mu}{b_0 + b_1 x + \cdots + b_\nu x^\nu}. \tag{5.9}$$

In this dissertation, we ignore the case where $b_0 = 0$ and assume that $\boldsymbol{b}(x)$ is normalized such that $b_0 = 1$. For a more detailed study of the case when $b_0 = 0$, the reader is referred to [BGM96] where a thorough investigation is made. The MacLaurin series of $[\mu/w]_{\boldsymbol{f}}(x)$ coincides with the power series of $\boldsymbol{f}(x)$ up to a degree of $\mu + \nu$. This can be mathematically expressed as follows

$$\deg\big(\boldsymbol{a}(x) - \boldsymbol{f}(x)\boldsymbol{b}(x)\big) > \mu + \nu \quad \text{as} \quad x \to 0. \tag{5.10}$$

To get a solution for $\boldsymbol{b}(x)$, Equation (5.10) can be expressed by the following linear system of equations

$$\begin{pmatrix} f_{\nu-\mu+1} & f_{\nu-\mu+2} & \cdots & f_\nu \\ f_{\nu-\mu+2} & f_{\nu-\mu+3} & \cdots & f_{\nu+1} \\ f_{\nu-\mu+3} & f_{\nu-\mu+4} & \cdots & f_{\nu+2} \\ \vdots & \vdots & & \vdots \\ f_\nu & f_{\nu+1} & \cdots & f_{\nu+\mu-1} \end{pmatrix} \begin{pmatrix} b_\mu \\ b_{\mu-1} \\ \vdots \\ b_1 \end{pmatrix} = - \begin{pmatrix} f_{\nu+1} \\ f_{\nu+2} \\ \vdots \\ f_{\nu+\mu} \end{pmatrix}. \tag{5.11}$$

After determining $\boldsymbol{b}(x)$, the polynomial $\boldsymbol{a}(x)$ is then calculated by

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_\nu \end{pmatrix} = \begin{pmatrix} f_0 & 0 & 0 & \cdots & 0 \\ f_1 & f_0 & 0 & \cdots & 0 \\ f_2 & f_1 & f_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ f_{\nu-1} & f_{\nu-1} & f_{\nu-2} & \cdots & f_0 \end{pmatrix} \begin{pmatrix} 1 \\ b_1 \\ b_2 \\ \vdots \\ b_\mu \end{pmatrix}. \tag{5.12}$$

If we were to calculate the Padé approximation of the syndrome $\boldsymbol{S}(x)$ as in (5.10), this would be equivalent to the Peterson algorithm [PW72]. Other decoding algorithms like BMA and EEA are also shown to be equivalent in [Fit95]. If the number of errors t is smaller than the decoding radius τ (or τ_p in the case of power decoding), the Padé approximation of $\boldsymbol{S}(x)$, represented in the key equation (3.9), is as follows

$$[(t-1)/t]_{\boldsymbol{S}}(x) = \frac{\boldsymbol{\Omega}(x)}{\boldsymbol{\Lambda}(x)}. \tag{5.13}$$

Having a number of independent equations more than or equal to the number of unknowns, we are able to determine the error locator polynomial $\boldsymbol{\Lambda}(z)$ by a Padé approximation. If $t > \tau$, the equations are not enough to calculate $\boldsymbol{\Lambda}(z)$; hence, the key equation (3.9) is unsolvable. It is still, however, possible to determine a low-degree Padé approximation of the syndrome $\boldsymbol{S}(x)$

$$[(\tau-1)/\tau]_{\boldsymbol{S}}(x) = \frac{\widehat{\boldsymbol{\Omega}}(x)}{\widehat{\boldsymbol{\Lambda}}(x)}, \tag{5.14}$$

where $\widehat{\boldsymbol{\Lambda}}(x)$ and $\widehat{\boldsymbol{\Omega}}(x)$ are low-degree approximations of the actual error locator $\boldsymbol{\Lambda}(x)$ and the actual error evaluator $\boldsymbol{\Omega}(x)$ respectively. Since $\mathbb{C}$ is algebraically closed, $\widehat{\boldsymbol{\Lambda}}(x)$ has a number of roots equal to its degree, which is equal to τ.

$$\widehat{\boldsymbol{\Lambda}}(x) = \prod_{i=1}^{\tau}(x-\beta_i), \quad \beta_i \in \mathbb{C}. \tag{5.15}$$

Although the roots of $\widehat{\boldsymbol{\Lambda}}(x)$ are close to those of $\boldsymbol{\Lambda}(x)$, they are not all (if any) the same roots. An example to show the closeness of the roots is presented in Figure 5.1. For a code $\mathcal{CRS}(50, 12)$ and an error with a locator polynomial $\boldsymbol{\Lambda}(x)$ with degree $t = 21$, the BMA is used to calculate the approximation $\widehat{\boldsymbol{\Lambda}}(x)$ with degree $\tau = 19$. The roots of both polynomials $\boldsymbol{\Lambda}(x)$ and $\widehat{\boldsymbol{\Lambda}}(x)$ are plotted.

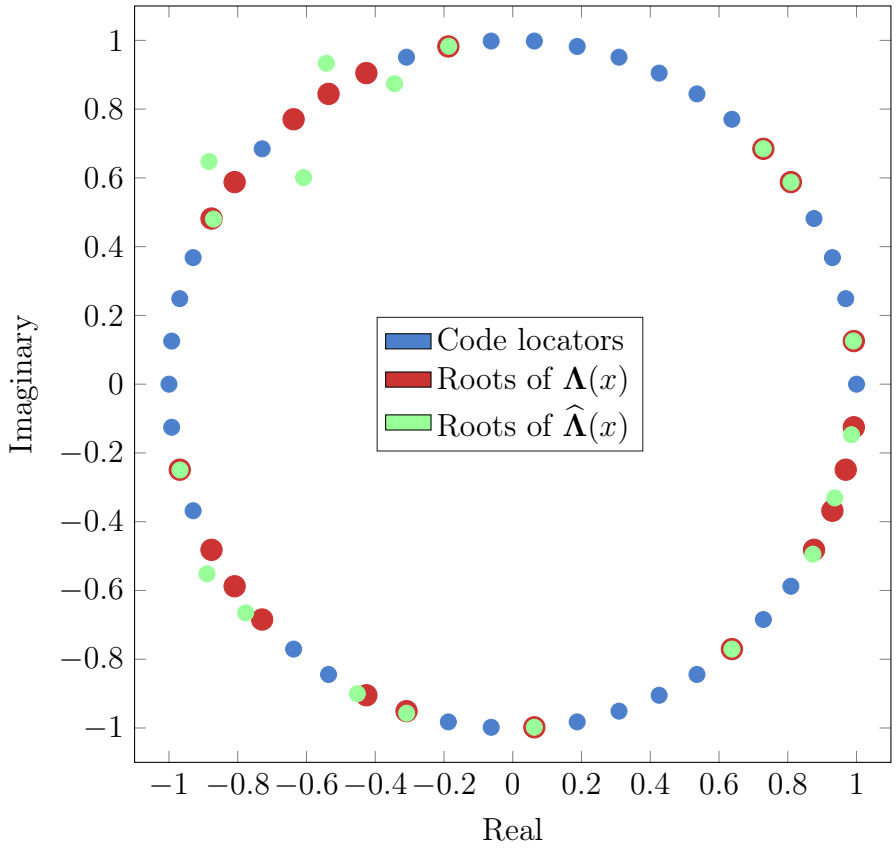

Figure 5.1: An example with code locators of a $\mathcal{CRS}(50, 12)$ code, roots of an error locator polynomial $\mathbf{\Lambda}(x)$ with $t = 21$ errors and the roots of its low-degree approximation $\widehat{\mathbf{\Lambda}}(x)$ with degree $\tau = 19$ calculated using BMA.

It is clear that although the number of errors t is greater than the radius τ, there are many roots of $\widehat{\mathbf{\Lambda}}(x)$ (9 out of 19) coinciding with roots of $\mathbf{\Lambda}(x)$. Even if they do not coincide, some are really close to the actual roots. Using this behaviour as motivation, it was shown in [Zö15], that $\widehat{\mathbf{\Lambda}}(x)$ can provide us with reliability information.

5.3.1 Reliability Information Calculation Methods

The quality of the reliability information provided by the low-degree Padé approximation not only depends on the degree difference between both $\mathbf{\Lambda}(x)$ and $\widehat{\mathbf{\Lambda}}(x)$ (which is $t - \tau$), but also on how it is calculated. Different decoding algo-

rithms, namely BMA, EEA and Peterson, are used and compared in [Zö15]. The reliability information provided by Peterson proved to be the best. However, in order to use power decoding (see Section 3.1.1) to provide a better low-degree approximation (since $\tau_p > \tau$), we intended to use either the variants of BMA or EEA, which are [SSB10] and [Nie16] respectively. Since [Zö15] shows that BMA is better than EEA in terms of reliability information, we only use the BMA and its power decoding variant from [SSB10] for the rest of the dissertation. Let the distance between the i-th code locator α^i and the j-th root of $\widehat{\mathbf{\Lambda}}(x)$ β_j be

$$\begin{aligned} \delta_{i,j} &= \left|(\alpha^i - \beta_j)\right| \\ &= d_{\mathrm{E}}(\alpha^i, \beta_j), \end{aligned} \tag{5.16}$$

where $i = 0, \ldots, n-1$ and $j = 1, \ldots, \tau$. The operation $|.|$ gives the magnitude of a complex-valued element. This is also equivalent to the Euclidean distance between the two points in a a 2-dimensional space (see Section 2.2.2). By utilizing these distances, we now propose different methods of calculating the reliability vector $\boldsymbol{\eta} = (\eta_0, \ldots, \eta_{n-1})$.

Direct Evaluation (Distance Product)

This method has been used in previous publications such as [MZB16] and [Zö15]. The polynomial $\widehat{\mathbf{\Lambda}}(x)$ is evaluated at all the code locators. Then, the magnitude of these evaluations is used as reliability of information.

$$\eta_i = |\widehat{\mathbf{\Lambda}}(\alpha^i)| = \left| \prod_{j=1}^{\tau} (\alpha^i - \beta_j) \right|. \tag{5.17}$$

Using the distance notation introduced earlier in (5.16), Equation (5.17) can be written as follows:

$$\eta_i = \prod_{j=1}^{\tau} \delta_{i,j}. \tag{5.18}$$

If this method is used, elements of $\boldsymbol{\eta}$ should be arranged in an ascending order with lower values corresponding to positions having a higher probability of being erroneous. The motivation for using this method is that it is considered to be a fast method, since it only requires evaluating the polynomial $\widehat{\mathbf{\Lambda}}(x)$ at the code locators and taking the magnitude. Figure 5.2 [Zö15] shows an example where this method is used to get the reliability information. In the example, the actual error locator polynomial $\mathbf{\Lambda}(x)$ has degree $t = 22$. The polynomial is calculated

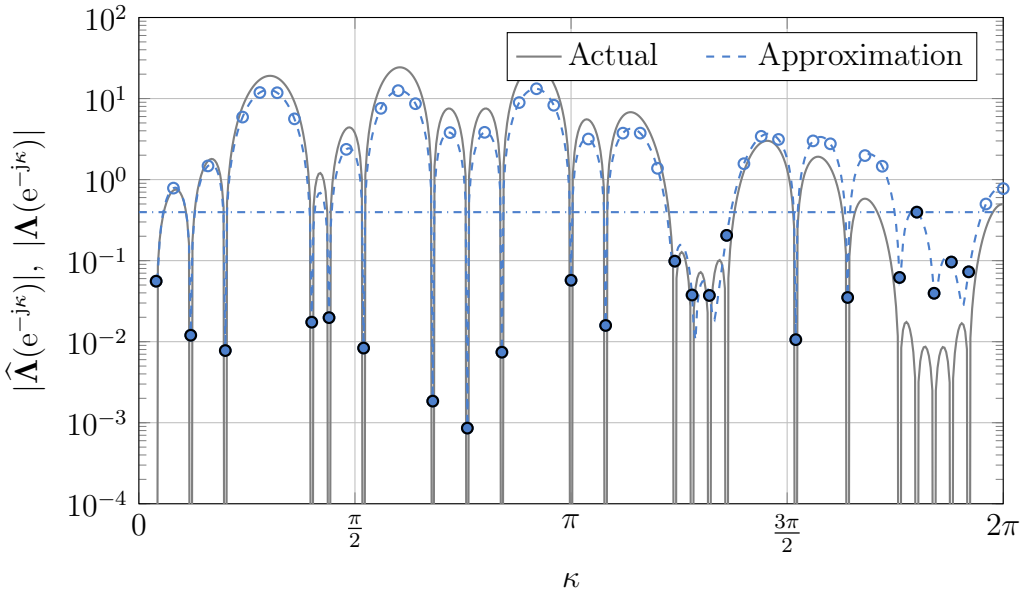

Figure 5.2: Magnitude of a low-degree approximation $|\widehat{\mathbf{\Lambda}}(z)|$ plotted as dashed line for $22 = t > \tau_p = 21$. Ideal reference $\mathbf{\Lambda}(z)$ is gray. Values corresponding to $\widehat{\mathbf{\Lambda}}(\alpha^i)$ are marked by circles. Filled marks belong to the τ smallest magnitudes, where a dash-dotted horizontal line indicates the largest of them [Zö15, MZB16].

using power decoding $\widehat{\mathbf{\Lambda}}(x)$ with $\tau_p = 21$ and is then evaluated at the unit circle ($e^{-j\kappa} = 1$). The magnitude of the polynomial at the code locators is marked by circles. It is shown that the locators with the least reliability value correspond to actual error locations (marked by filled circles).

Minimum Distance

In this method, the value η_i depends only on the minimum distance between a root β_i and the closest code locators.

$$\eta_i = \min_j \{\delta_{i,j}\}. \tag{5.19}$$

Unlike the first method, the reliability of a position i depends only on the closest root of $\widehat{\mathbf{\Lambda}}(x)$ to its locator α^i. Same as in the direct evaluation method, elements of $\boldsymbol{\eta}$ with lower values correspond to position with higher probability being erroneous.

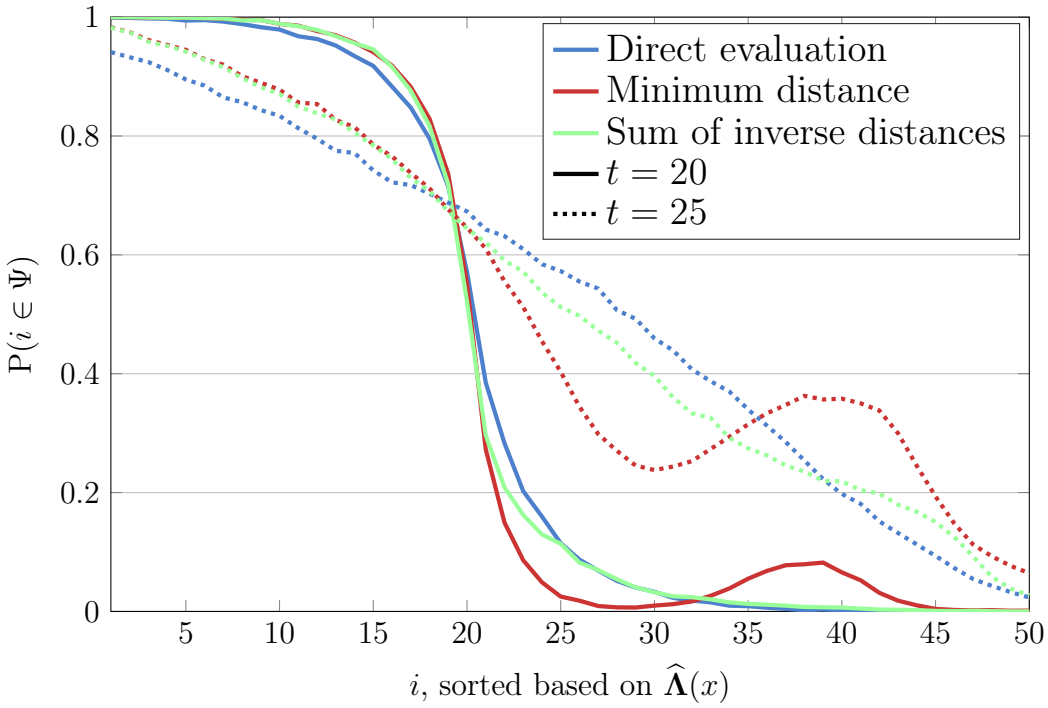

Figure 5.3: The probability of being an actual error position $\mathrm{P}(i \in \Psi)$. Positions are sorted based on reliability information extracted from $\widehat{\mathbf{\Lambda}}(x)$, which is calculated using BMA with $\tau = 19$ for a code $\mathcal{CRS}(50, 12)$ and number of errors $t = 20$ and 25.

Sum of Inverse Distances

Another way where all the roots of $\widehat{\mathbf{\Lambda}}(x)$ are taken into consideration is the sum of inverse distances method. Elements of the vector $\boldsymbol{\eta}$ are calculated as follows

$$\eta_i = \sum_{j=1}^{\tau} \frac{1}{\delta_{i,j}} \tag{5.20}$$

Since an inversion has been done, the elements of $\boldsymbol{\eta}$ should be arranged in a descending order such that higher values correspond to position with higher probability being erroneous.

In order to determine which of the methods better reflects the actual error positions, a numerical comparison is shown in Figure 5.3. Using BMA with a $\mathcal{CRS}(50, 12)$ code, we calculate the reliability vector $\boldsymbol{\eta}$ from the low-degree

approximation $\widehat{\mathbf{\Lambda}}(x)$ with degree $\tau = 19$. The number of errors used in the simulations are $t = 20, 25$. After sorting the positions according to $\boldsymbol{\eta}$, we calculate how often a position is an error ($\mathrm{P}(i \in \Psi)$). The main areas of interest are both extremes, which are the reliable positions (positions with low probability being erroneous) and the unreliable positions (positions with high probability of being erroneous). For the unreliable positions, both the minimum distance and the sum of inverse distances methods produce better reliability information than that provided by the direct evaluation method. As for the reliable positions, it is the other way around with the better reliability information being provided by the direct evaluation method. The minimum distance method turns out to be unstable in this part. This is because of the fact that the reliability of each position is only dependent on a single root of $\widehat{\mathbf{\Lambda}}(x)$ (the closest) resulting in an absence of a relation between the probability of error and the reliability information. Therefore, we conclude that the minimum distance method should not be used at all. Also, note that for a large number of errors, the direct evaluation and sum of inverse distances methods become almost the same.

Choosing which method of the two to use for reliability information extraction will in any case depend on the algorithm used. For example, algorithms depending on reliable positions being error-free to succeed such as the CAD algorithm from [Zö15] are not suited in combination with the sum of inverse distances method. A mixture of both methods could be used to provide better reliability information for both reliable and unreliable positions. This concept of mixing methods together has yet to be used and for the decoding algorithms presented in Section 5.4, the reliability information is calculated using only the direct evaluation method.

5.3.2 Properties of Reliability Information

While investigating the methods mentioned in Section 5.3.1, we came to realize that the reliability information acquired by all the methods had interesting properties, namely the rate dependence and the robustness against noise. For the simulations in this section the following applies

- The elements of the sparse error vector $\boldsymbol{e}$ (both real and imaginary) are extracted from a normally distributed random variable with mean zero and standard deviation $\sigma_e = 1/\sqrt{2}$.

- The reliability information is extracted using an approximated error locator polynomial $\widehat{\mathbf{\Lambda}}(x)$ which is calculated using the BMA.

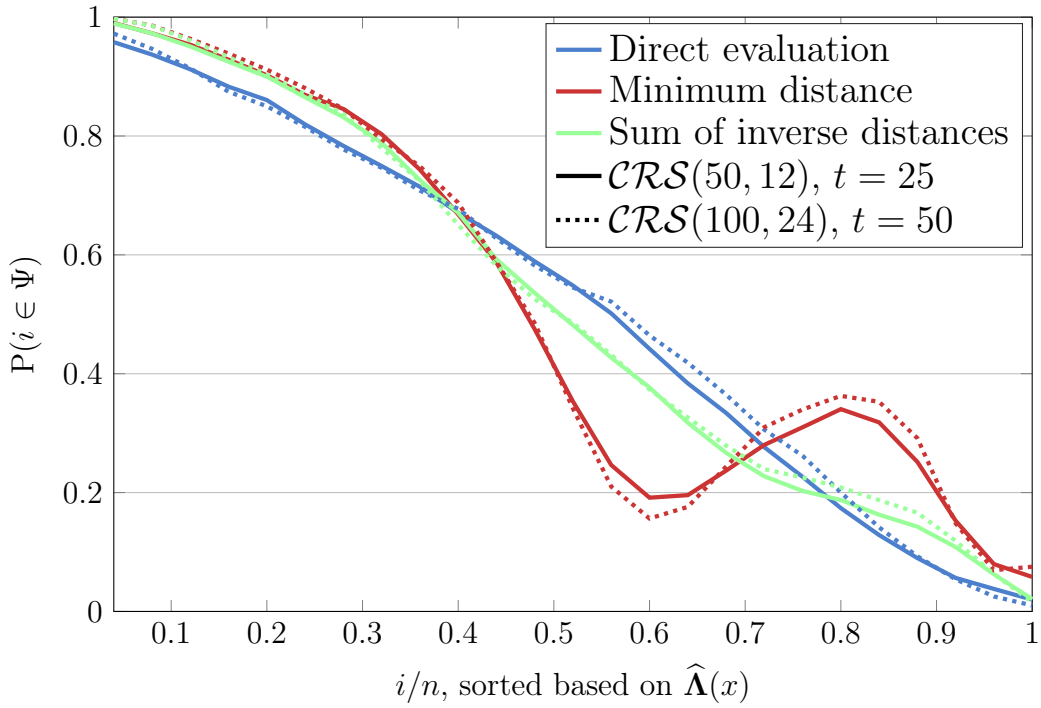

Figure 5.4: The probability of being an actual error position $\mathrm{P}(i \in \Psi)$, where the positions are sorted based on reliability information extracted from $\widehat{\mathbf{\Lambda}}(x)$. Codes having the same rate ($\mathcal{CRS}(50, 12)$, $\mathcal{CRS}(100, 50)$ are used with number of errors $t = 25$ and 50 respectively.

Rate Dependence

For the purpose of studying the effect of the length of the code n on the reliability information extracted, we performed a numerical simulation. The codewords used in the simulation are from a $\mathcal{CRS}(50, 12)$ code and a $\mathcal{CRS}(100, 24)$ code, where both codes have the same rate $R = k/n = 6/25$. Errors are added to codewords from both codes with a number of errors $t = 25$ and 50 respectively. The number of errors added are chosen such that the ratio $t/n = 1/2$ is equal for both cases. The results of this simulation are shown in Figure 5.4. From it we can observe that, for all three reliability information extraction methods, both codes produce the same behaviour, with some negligible variations.

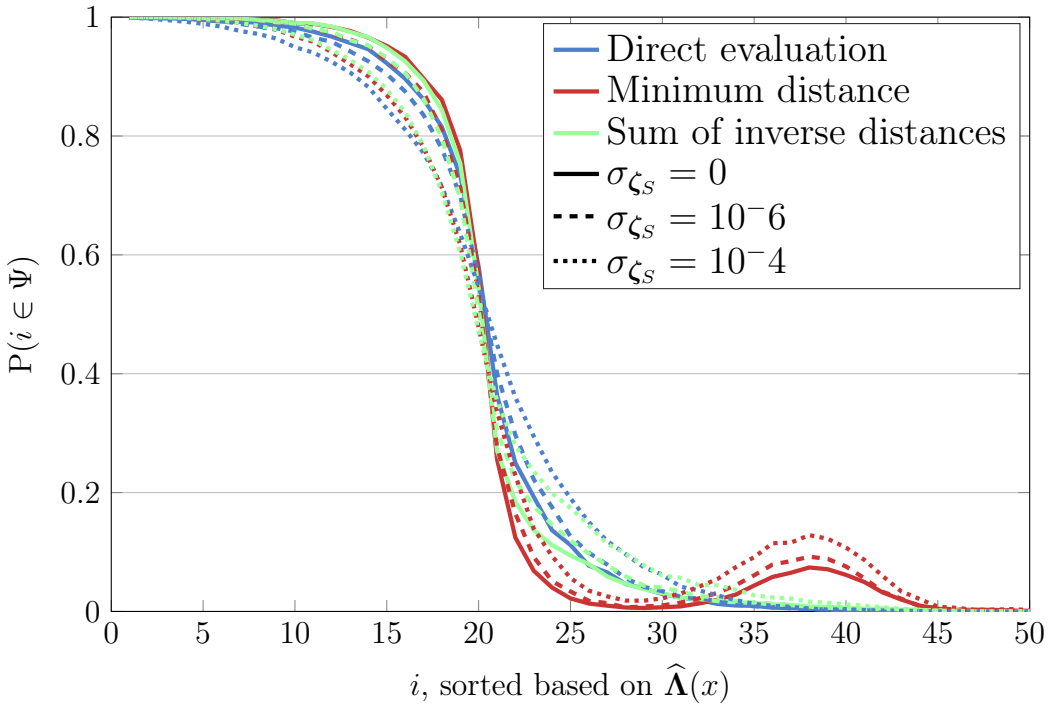

Figure 5.5: The probability of being an actual error position $\mathrm{P}(i \in \Psi)$, where the positions are sorted based on reliability information extracted from $\widehat{\mathbf{\Lambda}}(x)$ when syndrome $\boldsymbol{S}$ is affected by different noise levels. A code $\mathcal{CRS}(50, 12)$ is used with number of errors $t = 20$.

Robustness against Noise

Another factor we wished to study was the effect on noise on the reliability information on all three methods. As already mentioned in 5.1, there is always a noise term $\boldsymbol{\zeta}_S$ present in the syndrome (measurement) vector $\boldsymbol{S}$. Depending on the actual implementation (either hardware or software), this noise term represents the quantization errors and measurement noise, among other things of course. We assume that the elements of $\boldsymbol{\zeta}_S$ are extracted from a normally distributed random variable with mean zero and standard deviation σ_{ζ_S}. Using a code $\mathcal{CRS}(50, 12)$, we observe the behaviour of the reliability information under the effect of noise of different noise levels $\sigma_{\zeta_S} = 0$, 10^{-6} and 10^{-4}. Figures 5.5 and 5.6 show the cases when the number of errors is $t = 20$ and $t = 25$ respectively. It was expected that the noise would degrade the quality of the reliability information. However, we can see that this degradation is really small in Fig-

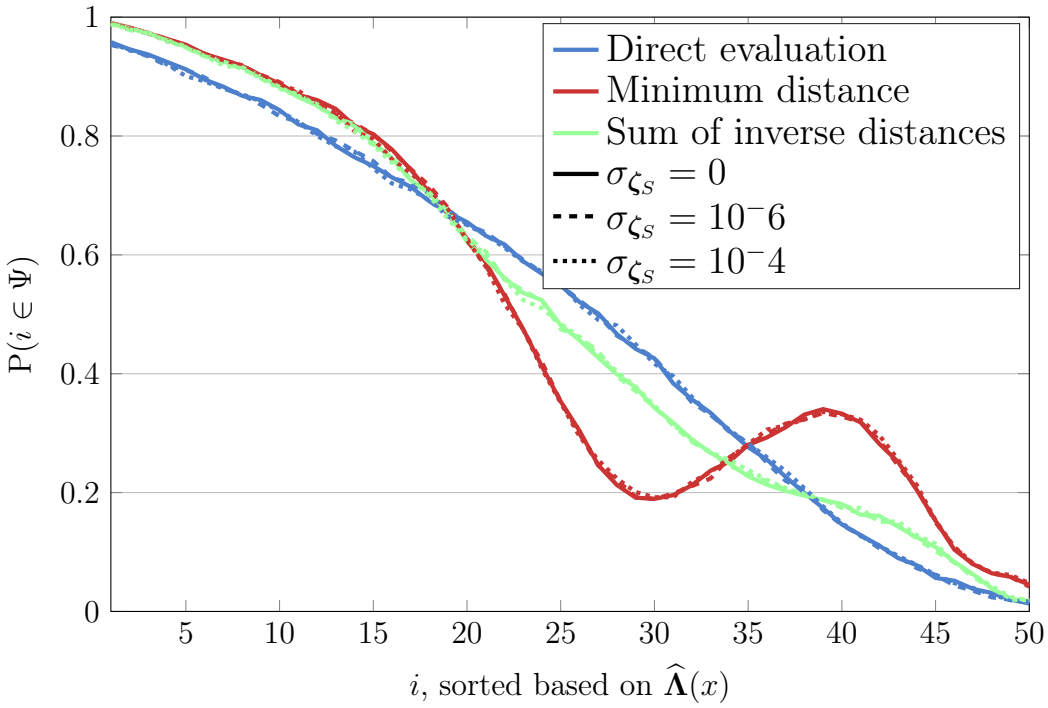

Figure 5.6: The probability of being an actual error position $\mathrm{P}(i \in \Psi)$, where the positions are sorted based on reliability information extracted from $\widehat{\mathbf{\Lambda}}(x)$ when syndrome $\boldsymbol{S}$ is affected by different noise levels. A code $\mathcal{CRS}(50, 12)$ is used with number of errors 25.

ure 5.5 and is not visible in 5.6. This implies that the reliability information obtained by the methods in question are all noise resistant. The only reason there would be a difference between the levels of degradation when dealing with different number errors is that the approximated polynomial $\widehat{\mathbf{\Lambda}}(x)$ contributes to this degradation, with it being dominant when the number of errors increase.

Results obtained in this section should come last if this chapter was chronologically ordered. That is why the algorithms discussed later in Section 5.4 use only reliability information obtained by the direct evaluation method. The effect of using the other two methods for reliability information calculation in the examined decoders is yet to be investigated.

5.4 Decoding Algorithms using Reliability

Although CRS codes were first presented many years ago in [Wol83] and [Mar84], there has been little investigation of reliability-based decoding algorithms for their decoding. There already exist many of those algorithms for RS codes over finite fields. However, not all of those algorithms are suitable to decode CRS in their original form. This is due to the presence of the noise factor $\boldsymbol{\zeta}$ in the vector $\boldsymbol{r}$ (or $\boldsymbol{\zeta}_S$ in the vector $\boldsymbol{S}$) as well as the quantization noise introduced with every computation done in the algorithm. When dealing with CRS codes it is important how an algorithm is implemented. It should be done in a way that keeps the noise from growing with each step in the algorithm. One would need to either change a few steps or even add new parts in the algorithm. This is not always possible to satisfy for all algorithms. A new criteria that should be considered when decoding CRS codes is the robustness against noise propagation through different phases of the algorithm used. Another reason for the shortage of research in this part is the fact that it is still a young branch. The reliability information obtained by the direct evaluation method shown in Section 5.3.1 was not known until [MRZB15] and [Zö15] were published. It is those publications that provided the framework upon which the algorithms discussed in this section are based.

5.4.1 Recursive Enhancement Algorithm

In this section, we discuss the Recursive Enhancement Algorithm (REA) introduced in [MZB16]. The REA focusses on enhancing the reliability information through the use of error/erasure decoding (see Section 3.3). Using the syndrome $\boldsymbol{S}(x)$ as input, one initially runs the BMA decoder. If the number of errors t is higher than half the minimum distance τ, the error locator $\boldsymbol{\Lambda}(x)$ is not obtainable with the BMA. Instead, the decoder obtains the low-degree approximation $\widehat{\boldsymbol{\Lambda}}(x)$. Then, the reliability vector $\boldsymbol{\eta}$ is calculated by the direct evaluation method as shown in Section 5.3.1. After that, erasures are introduced at some of the most unreliable positions based on the information contained in $\boldsymbol{\eta}$ and decoding is repeated. Let these positions be denoted by the set $\phi \subset \{0, \ldots, n-1\}$ and the erasure locator polynomial based on this set by $\boldsymbol{\Phi}(x)$. This time, the input for the BMA would be $\boldsymbol{\Phi}(x)\boldsymbol{S}(x)$, resulting in a new low-degree approximation $\overline{\boldsymbol{\Lambda}}(x)$. If it was guaranteed that the erased positions were in fact erroneous positions ($\phi \subset \Psi$), the new polynomial $\overline{\boldsymbol{\Lambda}}(x)$ would provide even better reliability vector $\overline{\boldsymbol{\eta}}$ for the rest of the positions. It is unfortunately not guaranteed, thus another approach is presented.

Rather than taking the risk and relying only on the new polynomial $\overline{\mathbf{\Lambda}}(x)$, the idea is to combine both $\widehat{\mathbf{\Lambda}}(x)$ and $\overline{\mathbf{\Lambda}}(x)$ to recalculate the reliability information, such that the direct evaluation is done on the multiplication of both polynomials $\widehat{\mathbf{\Lambda}}(x)\overline{\mathbf{\Lambda}}(x)$. Let the direct evaluation of $\mathbf{\Lambda}^*(x) = \widehat{\mathbf{\Lambda}}(x)\widetilde{\mathbf{\Lambda}}(x)$ produce the reliability vector $\boldsymbol{\eta}^*$. At this point, we can have one of three possible scenarios.

- $\phi \subseteq \Psi$: The vector $\boldsymbol{\eta}^*$ is an enhanced version of the vector $\boldsymbol{\eta}$. This event is the most frequent of all three.
- $\phi \cap \Psi \neq \emptyset$ and $\phi \not\subset \Psi$: If the number of error positions in the set ϕ are more than those which are not, the vector $\boldsymbol{\eta}^*$ is an enhanced version of the vector $\boldsymbol{\eta}$. Otherwise, it would be a degraded version. The probability that a degradation occurs is smaller than an enhancement.
- $\phi \cap \Psi = \emptyset$: The vector $\boldsymbol{\eta}^*$ is a degraded version of the vector $\boldsymbol{\eta}$. This event occurs with a very small probability.

The probability of each event occurring is solely dependent on the original reliability vector $\boldsymbol{\eta}$. However, if this vector is calculated using the direct evaluation method, the probability that an enhancement occurs is always higher than that of a degradation, regardless of the quality of the information contained in $\boldsymbol{\eta}$. This is a direct result of the behaviour observed when the reliability information extraction methods were analyzed in Section 5.3.1. Also, note that the use of other reliability extraction methods is not straight forward. That is because the multiplication of the low-degree approximations of $\mathbf{\Lambda}(x)$ would only be compatible to the direct evaluation method. One must change the way used to enhance the reliability vector $\boldsymbol{\eta}$ if other reliability extraction methods were to be used.

The core idea of the REA is to repeat this process multiple times while increasing the number of erased positions to enhance the reliability information even further. In some way, this is similar to GMD decoding explained in Section 3.3.2. The REA can be combined with any conventional error locator algorithm, not necessarily the BMA. Since the actual decoding process is unchanged one can also use either EEA or even PD for a better low-degree approximation of $\mathbf{\Lambda}(x)$. The output of the REA is the enhanced reliability information and is then conveyed to a CRS error evaluator algorithm, such as the CAD algorithm [Zö15, Algorithm 7.1].

The proposed REA is shown in Algorithm 5.1. The number of erased positions is increased by two with every enhancement iteration (starting from zero erased positions with up to $\lfloor (n-k)/2 \rfloor$ iteration). The complexity of Algorithm 5.1 is

Algorithm 5.1 Recursive Enhancement Algorithm (REA) [MZB16]

Input: Syndrome $\boldsymbol{S}(x)$, error locator algorithm $\boldsymbol{\Xi}$
Initialization: $\varepsilon \leftarrow 0$, $\boldsymbol{\Lambda}^*(x) \leftarrow 1$

while $\varepsilon < n - k$ **do**
  if $\varepsilon > 0$ **then**
    $\boldsymbol{\eta} \leftarrow$ Direct evaluation of $\boldsymbol{\Lambda}^*(x)$
    $\phi \leftarrow$ The most ε unreliable positions based on η
    $\boldsymbol{\Phi}(x) \leftarrow \prod_{i \in \phi}(x - \alpha^i)$
  else
    $\boldsymbol{\Phi}(x) \leftarrow 1$
  end if
  $\widehat{\boldsymbol{\Lambda}}(x) \leftarrow$ Run $\boldsymbol{\Xi}$ with input $\boldsymbol{\Phi}(x)\boldsymbol{S}(x)$
  $\boldsymbol{\Lambda}^*(x) \leftarrow \boldsymbol{\Lambda}^*(x) \cdot \widehat{\boldsymbol{\Lambda}}(x)$
  $\varepsilon \leftarrow \varepsilon + 2$
end while
$\boldsymbol{\eta} \leftarrow$ Direct evaluation of $\boldsymbol{\Lambda}^*(x)$
return $\boldsymbol{\eta}$

upper bounded by the complexity of the decoder used multiplied by the number of iterations.

Numerical Evaluation

In order to evaluate the performance of the REA, we ran a numerical simulation. The code chosen for the simulation is a $\mathcal{CRS}(50, 12)$ code. As error locator algorithms, we use the BMA and PD with decoding radii $\tau = 19$ and $\tau_p = 21$ respectively. As for the error evaluator algorithm, we chose CAD to produce the estimated error vector $\widehat{\boldsymbol{e}}$. The elements of the sparse error vector $\boldsymbol{e}$ and the noise vector $\boldsymbol{\zeta}_S$ (both real and imaginary parts) are extracted from a normally distributed random variable with mean zero with standard deviations of $\sigma_e = 1/\sqrt{2}$ and $\sigma = {}^{\sigma_{\zeta_S}}/\sqrt{2}$ with $\sigma_{\sigma_{\zeta_S}} = 10^{-5}$ respectively. The number of errors (sparsity) t has been varied from 19 to 32. In the simulation, a number of 10000 different error vectors are generated for each value of t. Each vector is compressed by the sensing matrix $\boldsymbol{H}$, corrupted by noise and then reconstructed using REA combined with CAD. The REA provides the CAD with reliability of information while the CAD calculates an estimation $\widehat{\boldsymbol{e}}$ of the sparse vector $\boldsymbol{e}$.

The level of performance is determined by observing two aspects: The quality

of the reliability information obtained from REA and the squared error $||\boldsymbol{e} - \widehat{\boldsymbol{e}}||^2$, which is equivalent to the square Euclidean distance $(d_{\mathrm{E}}(\boldsymbol{e}, \widehat{\boldsymbol{e}}))^2 = d_{\mathrm{E}}^2(\boldsymbol{e}, \widehat{\boldsymbol{e}})$. The performance of such a system is compared to that of OMP [CBL89], since it is one of the popular CS reconstruction algorithms used in the CS community. However, since we are dealing with a deterministic CS scheme where REA and CAD take advantage of the chosen sensing matrix structure, it is only fair to choose a sensing matrix for OMP from which it can benifit. Therefore, an optimized matrix based on Best Complex Antipodal Spherical Codes (BCASCs) [ZB15] is chosen as the sensing matrix for OMP, since they produce better performance when compared to random matrices [Zö15].

The results of the simulation are plotted using *boxplots*. These plots are used to show multiple properties of the distribution of a given dataset in one plot. There exist several different variants of boxplots [FHI89]. Here, we use the model provided by Tukey [Tuk77]. The most important properties are described as follows: The main part of the boxplot is built by a rectangle, which resembles the values between first and third quartile. The median is represented by a black horizontal bar within this box and the mean (average) is shown as a circle.

Let $\mathcal{Q}$ be the smallest set of positions sorted according to the reliability vector $\boldsymbol{\eta}$ such that all actual error positions are contained in this set ($\Psi \subseteq \mathcal{Q}$). The best case is when the first t unreliable positions (based on $\boldsymbol{\eta}$) are the actual error positions, such that $\Psi = \mathcal{Q}$. If the set $\mathcal{Q}$ is large, this means that there is at least one error position that falls far from the unreliable positions. In this case, it is safe to say that the reliability information contained in $\boldsymbol{\eta}$ is bad and does not reflect the location of actual errors any more. To investigate the quality of the reliability information provided by REA, we therefore investigate the size of the set $\mathcal{Q}$. Figure 5.7 contains boxplots illustrating the distribution of the cardinality of Q when reliability information is obtained through BMA and PD with REA for a different number of errors. For comparison, we show the previous best known reliability information which is that provided by PD without REA. From [Zö15, Section 7.3], we know that CAD is only able to reconstruct the vector $\boldsymbol{e}$ correctly if the errors are all located in the lowest $n - k = 38$ positions. Therefore, if the set $\mathcal{Q}$ has a cardinality falling in the greyed out areas, CAD will fail if given the vector $\boldsymbol{\eta}$.

As shown in Figure 5.7, PD without REA starts to have outliers in the grey area starting from $t = 22$. This is expected since its correction radius is restricted to $\tau_p = 21$. This means that the quality of the reliability information it provides starts to deteriorate when the number of errors is larger than $\tau_p = 21$. However, when using REA, outliers do not start to appear in the grey area until $t = 25$. The performance of both variants of REA could be considered (to some extent)

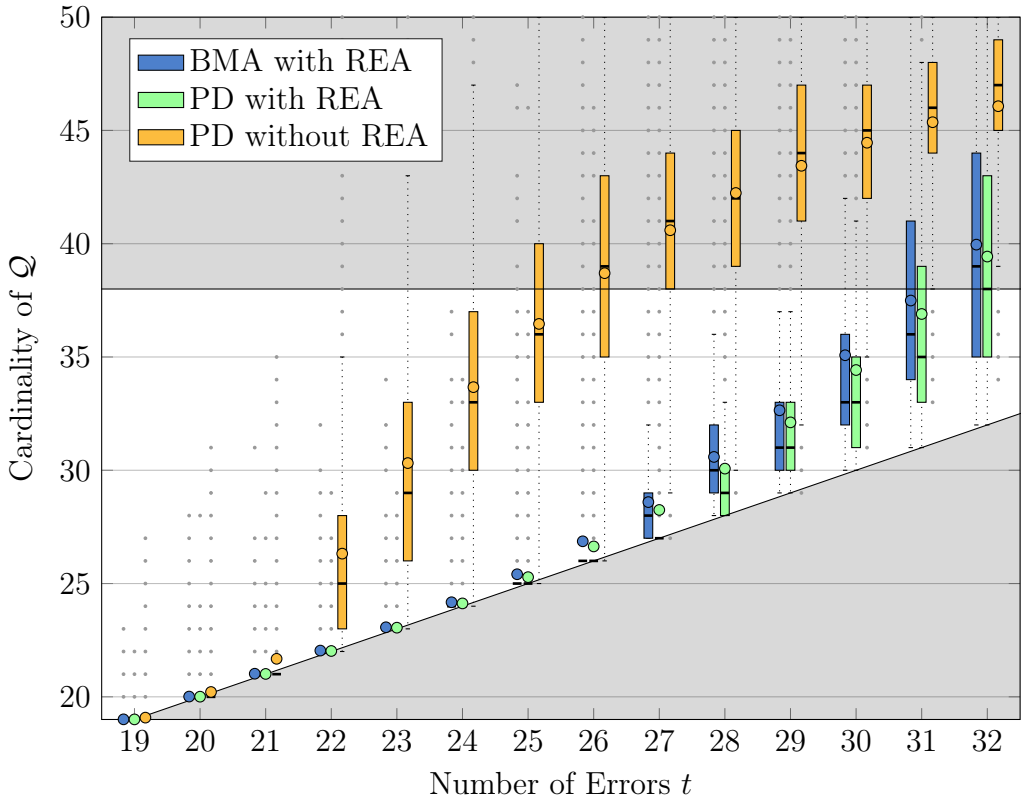

Figure 5.7: Boxplots illustrating the distribution of the cardinality of $\mathcal{Q}$, where $\mathcal{Q}$ the smallest set of positions sorted according to $\boldsymbol{\eta}$ such that all actual error positions are contained in this set. Non-gray area allows application of CAD [MZB16].

similar. But one should note that there are more outliers for BMA with REA than for PD with REA, which is still not at all strange since the decoding radius of BMA ($\tau = 19$) is lower than PD ($\tau_p = 21$).

The rectangular box representing the area between the first and third quartile does not appear until the number of errors is $t = 26$ for BMA with REA and $t = 27$ for PD with REA. This means that for more than 50% of the time the first t unreliable positions are the actual error locations. This does not continue for long with the quality of the reliability information starting to deteriorate. This figure gives us a kind of a prediction on the performance of CAD when combined with the reliability information obtained from the simulated methods.

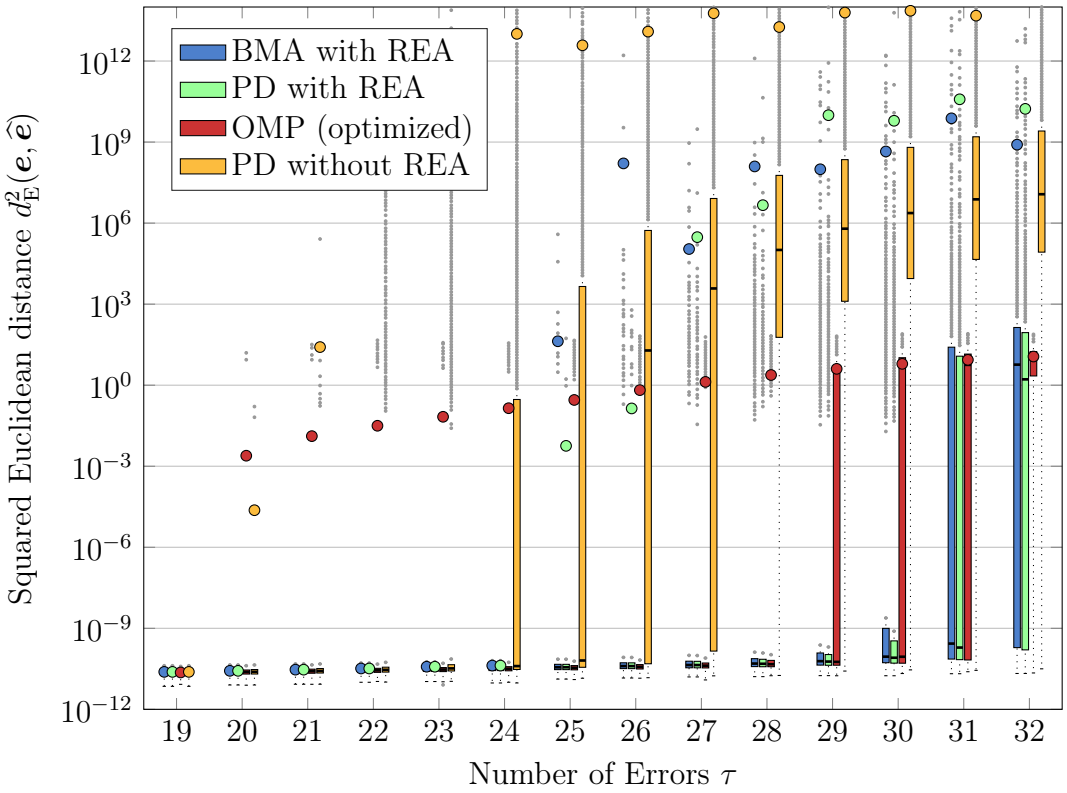

Figure 5.8: Boxplots illustrating the distribution of the squared Euclidean distance $d_{\mathrm{E}}^2(\boldsymbol{e}, \widehat{\boldsymbol{e}})$ for a deterministic CRS based CS scheme with different algorithms and the OMP algorithm in combination with an optimized sensing matrix [MZB16].

In Figure 5.8, the distribution of the squared Euclidean distance $d_{\mathrm{E}}^2(\boldsymbol{e}, \widehat{\boldsymbol{e}})$ resulting from the chosen reconstruction algorithms is illustrated by boxplots. Lets focus first on the CRS reconstruction methods. Having an enhanced reliability information for the CAD algorithm provided a considerable gain to that without enhancement. PD without REA already starts to have outliers starting from $t = 20$ and falls apart producing an estimation $\widehat{\boldsymbol{e}}$ that is far from the original vector $\boldsymbol{e}$ as the number t increases. While the methods that used enhancement (both BMA/PD with REA) produce no outliers until the number of errors reaches $t = 24$. That means that the enhancement allows (perfect) reconstruction of the vector $\boldsymbol{e}$ even when the number of errors is beyond the decoding radius of the original algorithm. As the number of errors increase, the reconstruction rates start to drop, but up to $t = 30$ more than 50% of the

reconstruction attempts with enhancement succeed. By this we also show that the new proposed methods also outperform the optimized OMP. The optimized OMP has already outliers starting from $t = 20$ (similar to PD without REA) and becomes totally unreliable at $t = 29$, while both REA variants start being so at $t = 31$.

From these results, we can see that the REA utilizes the concept of error/erasure decoding to provide an improved reliability information, which allows the reconstruction of the sparse error vector in scenarios beyond the classical decoding radius. It is even able to outperform the optimized OMP algorithm for reasonable noise levels. All this is achieved with the non-optimized Algorithm 5.1, which is a GMD-like version. There is still room for improvement by changing the number of erasures in the first iteration as well as the increase of erasures in every other iteration. These parameters will definitely have a direct effect on the performance, thus, they should be further studied for an optimized application of the algorithm.

The reliability information is utilized by the novel REA in order to provide an improved approximation of the error locator polynomial for scenarios well beyond the classical correction radius. REA is based on fundamental ideas of error/erasure decoding. Simulations showed that the new algorithm not only allows improving significantly on the performance but also surpasses the OMP algorithm with optimized low-coherence matrices for reasonable noise levels.

5.4.2 Guruswami–Sudan-based Generalized Minimum Distance Decoding

In this section, we investigate the application of the GS algorithm (see Section 3.2.1) for CRS codes with the help of reliability information. Unlike the REA in the previous section, this algorithm is interpolation-based. The reconstruction of the sparse error vector $\boldsymbol{e}$ is not directly done using the syndrome $\boldsymbol{S}$. Instead, we first calculate the vector $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$ using (5.7). Using the elements of $\boldsymbol{r}$ as interpolation points, we then recover the IDFT of the arbitrary codeword $\boldsymbol{c} \in \mathcal{CRS}$ by a root finding algorithm, such as the RR algorithm (see Algorithm 3.2). Finally, vector $\boldsymbol{e}$ is simply calculated by subtracting $\boldsymbol{c}$ from $\boldsymbol{r}$.

In theory, this decoding process is applicable to CRS codes as it is for RS codes over finite fields. Having a decoding radius larger than that in classical decoders ($\tau_{GS} \geq \tau$), such as BMA, was motivation for us to pursue its application in CS sparse reconstruction. However, since we are dealing with a new setup, namely the use of complex-valued numbers, the stability of the GS algorithm proved to

be a problem. The numerical inaccuracies arising from floating point calculations proved catastrophic for the application of the GS algorithm for CRS-based CS. It was first to be investigated by Parvaresh and Hassibi in [PH08], where they indicated the sensitivity of the RR root finding algorithm to numerical inaccuracies. Therefore, to be able to use these algorithms, each phase in the decoding process (interpolation and root finding) should be tweaked and adjusted to the new environment we are dealing with. Also by making use of the obtainable reliability information (see Section 5.3.1), we are able to construct a better algorithm which is more efficient in dealing with these numerical inaccuracies. The findings and results of this part of the dissertation can also be found in [MPB17].

First, let us discuss the interpolation step. Just to remind the reader, the interpolation step is concerned with the calculation of the polynomial $\boldsymbol{Q}(x, y)$.

$$\boldsymbol{Q}(x, y) = \boldsymbol{Q}_0(x) + \boldsymbol{Q}_1(x)y + \boldsymbol{Q}_2(x)y^2 + \cdots + \boldsymbol{Q}_\ell(x)y^\ell. \tag{5.21}$$

This polynomial should satisfy the conditions mentioned in Theorem 3.2, such that $(y - \boldsymbol{C}(x))$ divides it. The polynomial $\boldsymbol{C}(x)$ is the IDFT of the codeword $\boldsymbol{c}(x)$ (see Definition 5.2). For the interpolation to succeed, we choose the algorithms parameters to satisfy the following inequality

$$s(\ell + 1)(n - \tau_{GS}) - \frac{1}{2}\ell(\ell + 1)(k - 1) - 1 > n\binom{s+1}{2}, \tag{5.22}$$

where s and ℓ are the multiplicity and the list size. By solving a linear system of equations constructed from the interpolation points (α^i, r_i) for $i = 0, \ldots, n-1.$, we are able to calculate the coefficients of the polynomials $\boldsymbol{Q}_i(x)$ for $i = 0, \ldots, \ell$ with $s(n - \tau_{GS}) - i(k - 1)$ coefficients each. From this step, the numerical inaccuracies start to appear in the decoding process. Since our goal is to minimize it, we solve the linear system of equations using the Singular Value Decomposition (SVD). According to [DR08, Section 4.7], [GVL96, Section 2.5], the SVD is considered to be a stable method. As a result, the interpolation step can be considered the most stable step in the algorithm. There are other dedicated methods for the interpolation step such as [Ale05, Tri07, BB10, ZGA11]. Although these methods provide a more efficient interpolation, their numerical stability over $\mathbb{C}$ are yet to be investigated.

After getting $\boldsymbol{Q}(x, y)$, the next phase is to obtain the root $(y - \boldsymbol{C}(x))$ using a root finding algorithms. When dealing with polynomials over a finite field, the RR algorithm is one of the most efficient root finding algorithms. Still, it can not be used in its original form to find complex-valued roots. So we introduce a few changes and end up with its modified version, mRR, shown in Algorithm 5.2.

Algorithm 5.2 modified Roth-Ruckenstein (mRR) [MPB17]

Input: Bivariate polynomial $\boldsymbol{Q}(x, y)$, dimension k, and $i \in \mathbb{N}$
Global Variables: Set $\mathcal{U} \subseteq \mathbb{C}_k[x]$
Polynomial $\boldsymbol{f}(x) \in \mathbb{C}_k[x]$

1: **if** $i = 0$ **then**
2: $\quad \mathcal{U} = \emptyset$
3: **end if**
4: $\widetilde{\boldsymbol{Q}}(x, y) \leftarrow \boldsymbol{Q}(x, y)$
5: **if** $\widetilde{\boldsymbol{Q}}_{i,j} < \epsilon$ **then**
6: $\quad \widetilde{\boldsymbol{Q}}_{i,j} \leftarrow 0$
7: **end if**
8: $\omega \leftarrow$ largest integer such that x^ω divides $\widetilde{\boldsymbol{Q}}(x, y)$
9: $\boldsymbol{T}(x, y) \leftarrow x^{-\omega}\widetilde{\boldsymbol{Q}}(x, y)$
10: $\mathcal{Z} \leftarrow$ set of all distinct y-roots of $\boldsymbol{T}(0, y)$ in $\mathbb{C}$
11: **for** each $\gamma \in \mathcal{Z}$ **do**
12: $\quad f_i \leftarrow \gamma$
13: $\quad$ **if** $i < k - 1$ **then**
14: $\quad\quad$ mRR$(\boldsymbol{T}(x, xy + \gamma), k, i + 1)$
15: $\quad$ **else**
16: $\quad\quad \mathcal{U} \leftarrow \mathcal{U} \cup \{\boldsymbol{f}(x)\}$
17: $\quad$ **end if**
18: **end for**

Output: The list of y-roots $\mathcal{U}$

We introduced two main modifications (shown in red in Algorithm 5.2): The first is the introduction of a threshold ϵ before finding the integer ω. Since we can never end up with an exact zero during floating point calculations we must force small values to zero. The second is the removal of the IF condition in line 12 in Algorithm 3.2. This IF condition is to make sure that a solution is a y-root of $\boldsymbol{Q}(x, y)$. But due to the fact that there will always be a deviation from the correct solution, it does not make sense to keep it. This will unfortunately allow more wrong solutions to be present in the set $\mathcal{U}$. This problem will be addressed shortly in a process that refines the entries in the output set. First, we explain how numerical inaccuracies destabilize this algorithm, making it unsuitable for operation over the complex field.

To find the polynomial $\boldsymbol{f}(x) = f_0 + \cdots + f_{k-1}x^{k-1}$, the algorithm calculates its coefficients one by one, starting with the calculation of f_0. With every iter-

ation, the algorithm recursively calculates a coefficient f_i using the previously calculated coefficient f_{i-1}. The process is then repeated until all the coefficients are found. Assuming a small insignificant numerical inaccuracy occurred when calculating a coefficient f_i, this discrepancy propagates as we calculate every new coefficient, which will eventually add up, increasing exponentially, as the algorithm progresses. That is why the most accurately calculated coefficient is f_0, while the high order coefficients can sometimes be fatally inaccurate.

That's not the only part where numerical inaccuracies are produced. Another even more critical element to be calculated is the integer ω, which is the greatest integer such that x^ω divides $\widetilde{\boldsymbol{Q}}(x, y)$. The process of calculating ω is done by checking if the polynomials $\widetilde{\boldsymbol{Q}}_i(x)$ for $i = 1, \ldots, \ell$ are divisible by x^ω, hence, the first ω coefficients are zero. If a coefficient of $\widetilde{\boldsymbol{Q}}_i(x)$ happens to be slightly above the threshold ϵ, the integer ω will be wrongly calculated. If this event occurred in some iteration i, the coefficient f_i will also be wrong, causing all the following coefficients of $\boldsymbol{f}(x)$ to be totally inaccurate. To sum things up, we can say that the numerical inaccuracies are inevitable and they are more visible in high order coefficients of $\boldsymbol{f}(x)$. It is indisputable that with our modification (removal of the IF condition from the RR algorithm) and the numerical inaccuracies, the set $\mathcal{U}$, which is output of the mRR algorithm, will include degraded versions of the correct solutions as well as wrong ones. Another step to refine the elements of the set $\mathcal{U}$ must be introduced. We propose the use of Newton's method for the refinement step.

Newton's method

Newton's method is used to numerically find a root of any polynomial [DR08]. It can be also used to find an approximate solution of a non-linear system of equations, which is the same as finding the roots of continuously differentiable functions. Our goal is to find the root $\boldsymbol{f}(x)$ with degree $k-1$ for the polynomial $\boldsymbol{Q}(x, y)$.

$$\boldsymbol{Q}(\alpha^i, \boldsymbol{f}(\alpha^i)) = 0 \quad \forall i = 0, \ldots, n-1. \tag{5.23}$$

This can be done by considering the following evaluation map as a function in the coefficients $f_0, \ldots, f_{k-1}$ of $\boldsymbol{f}(x)$,

$$\begin{gathered} \varphi : \mathbb{C}^k \to \mathbb{C}^n, \\ \boldsymbol{f} := \begin{pmatrix} f_0 \\ \vdots \\ f_{k-1} \end{pmatrix} \mapsto \begin{pmatrix} \varphi_1(\boldsymbol{f}) \\ \vdots \\ \varphi_n(\boldsymbol{f}) \end{pmatrix} = \begin{pmatrix} \boldsymbol{Q}(\alpha^0, \boldsymbol{f}(\alpha^0)) \\ \vdots \\ \boldsymbol{Q}(\alpha^{n-1}, \boldsymbol{f}(\alpha^{n-1})) \end{pmatrix}, \end{gathered} \tag{5.24}$$

and then solving a system of k non-linear equations

$$\varphi(\boldsymbol{f}) = \mathbf{0}. \tag{5.25}$$

Given an initial point $\boldsymbol{z}_0 \in \mathbb{C}^k$, Newton's method tries to converge to an actual solution $\boldsymbol{z} \in \mathbb{C}^k$ of $\varphi(\boldsymbol{z}) = \mathbf{0}$. This is done iteratively with an iteration denoted by $\boldsymbol{z}_{i-1} \mapsto \boldsymbol{z}_i$. In a single iteration, a new vector $\boldsymbol{z}_i$ is caculated by solving the following linear system of equations

$$(\boldsymbol{z}_i - \boldsymbol{z}_{i-1}) \cdot \boldsymbol{J}_\varphi(\boldsymbol{z}_{i-1}) = -\varphi(\boldsymbol{z}_{i-1}), \tag{5.26}$$

where $\boldsymbol{J}_\varphi(\boldsymbol{z}_{i-1})$ is the Jacobi matrix of φ at the point $\boldsymbol{z}_{i-1}$,

$$\boldsymbol{J}_\varphi(\boldsymbol{z}_{i-1}) = \begin{pmatrix} \frac{\partial \varphi_1}{\partial f_0}(\boldsymbol{z}_{i-1}) & \dots & \frac{\partial \varphi_1}{\partial f_{k-1}}(\boldsymbol{z}_{i-1}) \\ \vdots & \ddots & \vdots \\ \frac{\partial \varphi_n}{\partial f_0}(\boldsymbol{z}_{i-1}) & \dots & \frac{\partial \varphi_n}{\partial f_{k-1}}(\boldsymbol{z}_{i-1}) \end{pmatrix} \in \mathbb{C}^{n \times k}. \tag{5.27}$$

To calculate the derivative $\frac{\partial \varphi_i}{\partial f_j}(\boldsymbol{f})$, let us look first at the exact expression of $\varphi_i(\boldsymbol{f})$ for $i = 0, \dots, n-1$,

$$\varphi_i(\boldsymbol{f}) = \sum_\mu \sum_\nu Q_{\mu,\nu} (\alpha^i)^\mu (\boldsymbol{f}(\alpha^i))^\nu \tag{5.28}$$

$$= \sum_\mu \sum_\nu Q_{\mu,\nu} \alpha^{i\mu} \left(\sum_{\xi=0}^{k-1} f_\xi \alpha^{i\xi} \right)^\nu, \tag{5.29}$$

where the indices μ, ν depend on the the degree restrictions provided by parameters used in the GS interpolation problem. The only part that is relevant to the derivation is the expression $(\boldsymbol{f}(\alpha^i))^\nu$, therefore the derivative is as follows

$$\frac{\partial \varphi_i}{\partial f_j}(\boldsymbol{f}) = \sum_\mu \sum_\nu Q_{\mu,\nu} \nu \alpha^{i(\mu+j)} \left(\sum_{\xi=0}^{k-1} f_\xi \alpha^{i\xi} \right)^{\nu-1}. \tag{5.30}$$

Equation (5.30), gives us an explicit expression of the Jacobi matrix $\boldsymbol{J}_\varphi(\boldsymbol{f})$ for any $\boldsymbol{f} \in \mathbb{C}^k$, allowing us to proceed with solving the system of equations represented by (5.26).

From the root finding step, we already have a set of polynomials $\mathcal{U}$ that contains degraded versions of roots of the polynomial $\boldsymbol{Q}(x, y)$. Using these polynomials as initial points to Newton's method, we try to find a more accurate

approximation for the roots of $\boldsymbol{Q}(x, y)$. The number of iterations needed to reach the correct solution, depends on the initial point used as well as the behaviour of the polynomial $\boldsymbol{Q}(x, y)$. It might even happen that the method may not converge to a root at all, regardless of how many iterations were to be considered. However, if it does, it often locally converges quadratically in the number of iterations. In order to achieve convergence, a "good" initial point $\boldsymbol{z}_0$ must be chosen. As already mentioned, the numerical inaccuracies occurring in a polynomial $\boldsymbol{p}(x) \in \mathcal{U}$ is at its lowest in the coefficient p_0 and at its highest in the coefficient p_{k-1}. As a result to this fact, we decided to choose the initial point to be $\boldsymbol{z}_0 = p_0 + \cdots + p_l x^l$, where $l < k-1$. By ignoring the coefficients that contain higher inaccuracies, we end up with better convergence rate (smaller number of iterations). The effect of the parameter l on the whole decoding process is yet to be investigated. For the rest of this section, we choose $l = \lfloor k/2 \rfloor$. Later numerical simulations show that this choice provides a good initial point.

At this stage, a list of possible roots to the polynomial $\boldsymbol{Q}(x, y)$ has been obtained. However, since the algorithm is to be used for the application of CS, it is expected to have a distinct output rather than a list. To acquire a single solution, an extra operation is to be executed. Aided by the possibility of calculating reliability information for CRS codes and error/erasure decoding, we propose a method to obtain a single solution allowing us to reconstruct the sparse vector $\boldsymbol{e}$.

GS-based GMD decoding

In order to choose the best solution from the list obtained by Newton's method, we propose a multi trial GMD decoder based on the GS algorithm. Similar to classical GMD decoding, within each trial erasures are introduced at the unreliable elements of the vector $\boldsymbol{r}$ depending on some reliability information provided to the algorithm. There exists many different definitions for an erasure, each depending on the decoding method used. Here, an erasure is defined as an element in the vector $\boldsymbol{r}$ which is not used in the interpolation step in the GS algorithm. For each trial, the number of erased positions is increased, which implies that a new polynomial $\boldsymbol{Q}(x, y)$ is calculated for each decoding trial using the unerased elements of $\boldsymbol{r}$. The parameters of the interpolation ℓ and s are recalculated for each trial since the number of interpolation points changes. Their values are chosen to be as minimal as possible while satisfying (5.22). Let the interpolation step be denoted by $GS(\boldsymbol{r}, \mathcal{I})$, where the set $\mathcal{I} \in \{0, \ldots, n-1\}$ contains the position of the erased elements. In a decoding trial, the roots of the calculated $\boldsymbol{Q}(x, y)$ are obtained with the mRR (Algorithm 5.2) and refined using

Newton's method; providing us with a list of possible solutions for the vector $\boldsymbol{e}$. Each entry in this list gets a score which depends on the number of iterations in which it appeared. After finishing all the decoding trials, the entry with the highest score is considered as the solution of reconstruction of the vector $\boldsymbol{e}$.

Algorithm 5.3 GS-based GMD decoding [MPB17]

Input: Vector $\boldsymbol{r}$, length n, dimension k, decoding radius τ_g, number of trials N, initial erasures ε and erasure step ε^+

Initialization: $i \leftarrow 0$, $\mathcal{L} \leftarrow \{\}$

$\boldsymbol{\Lambda}(x) \leftarrow \text{BMA}(\boldsymbol{r})$ # Classical decoding
if $\boldsymbol{\Lambda}(x)$ is a proper error locator **then**
  $\widehat{\boldsymbol{e}} \leftarrow \text{GZ}(\boldsymbol{r}, \boldsymbol{\Lambda}(x))$
else # GS-based GMD decoding
  $\boldsymbol{\eta} \leftarrow$ Direct evaluation of $\boldsymbol{\Lambda}(x)$ # Reliability information
  while $i < N$ **do**
    $\mathcal{I} \leftarrow$ Unreliable to be erased positions (based on $\boldsymbol{\eta}$)
    $\tau_{GS} \leftarrow \tau_g - \varepsilon$, $n \leftarrow n - \varepsilon$
    Choose ℓ, s such that Equation (5.22) is satisfied.
    $\boldsymbol{Q}(x, y) \leftarrow \text{GS}(\boldsymbol{r}, \mathcal{I})$ # Interpolation
    $\mathcal{U} \leftarrow \text{mRR}(\boldsymbol{Q}(x, y), k, 0)$ # Root finding
    for each $\boldsymbol{U} \in \mathcal{U}$ **do**
      $\widetilde{\boldsymbol{C}} \leftarrow \text{Newton}(\boldsymbol{U})$ # Newton's method
      $\widetilde{\boldsymbol{c}} \leftarrow DFT(\widetilde{\boldsymbol{C}})$
      $\widetilde{\boldsymbol{e}} \leftarrow \boldsymbol{r} - \widetilde{\boldsymbol{c}}$
      if $\text{supp}(\widetilde{\boldsymbol{e}} < \epsilon) \leq \tau_g$ **then**
        if $\widetilde{\boldsymbol{e}} \in \mathcal{L}$ **then**
          $\text{score}(\widetilde{\boldsymbol{e}}) \leftarrow \text{score}(\widetilde{\boldsymbol{e}}) + 1$
        else
          $\mathcal{L} \leftarrow \mathcal{L} \cup \widetilde{\boldsymbol{e}}$, $\text{score}(\widetilde{\boldsymbol{e}}) \leftarrow 1$ # New entry in $\mathcal{L}$
        end if
      end if
    end for
    $i = i + 1$, $\varepsilon \leftarrow \varepsilon + \varepsilon^+$
  end while
  $\widehat{\boldsymbol{e}} \leftarrow \underset{\boldsymbol{l} \in \mathcal{L}}{\text{argmax}}\ \text{score}(\boldsymbol{l})$ # Solution with highest score
end if

Output: The reconstructed sparse error vector $\widehat{\boldsymbol{e}}$

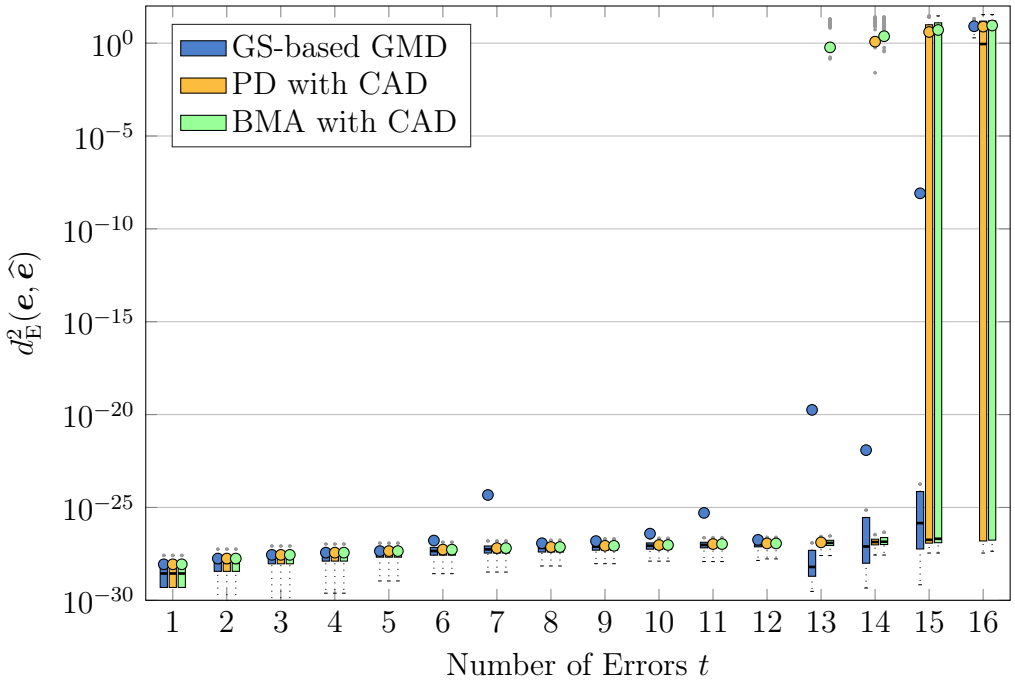

Figure 5.9: Boxplots illustrating the distribution of the squared Euclidean distance $d_{\mathrm{E}}^{2}(\boldsymbol{e}, \widehat{\boldsymbol{e}})$ in a noiseless scenario for a $\mathcal{CRS}(32, 8)$ code for different decoding schemes.

For this decoding setup, we define a few parameters: Let the number of decoding trials be denoted by N, the algorithms designed decoding radius by τ_g, the initial number of erasures by ε and the increase in the number of erasures per trial by ε^{+}. These parameters should be chosen depending on the desired performance and complexity of the algorithm. Before running this algorithm, decoding using a classical BMA decoder is attempted. If it fails, the output of the BMA is used to calculate reliability information using the direct evaluation method mentioned in Section 5.3.1. The proposed GS-based GMD decoder is shown in Algorithm 5.3.

Numerical Evaluation

Numerical simulations are performed to investigate the performance of Algorithm 5.3. To provide a better insight into its performance, we compare the algorithm with state-of-the-art CRS decoding algorithms, namely the BMA and PD with CAD from [Zö15]. The codes used in the simulations are the $\mathcal{CRS}(32, 8)$

and $\mathcal{CRS}(16, 4)$ codes. For each code, we randomly generate a total number of 10000 sparse error vectors $\boldsymbol{e}$ with varying number of non-zero elements t to be added to a randomly generated codeword $\boldsymbol{c}$ resulting in a vector $\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$. The vector $\boldsymbol{r}$ is affected by a randomly generated noise vector $\boldsymbol{\zeta}$. The elements of a sparse error vector $\boldsymbol{e}$ and the noise vector $\boldsymbol{\zeta}$ (both real and imaginary parts) are extracted from a normally distributed random variable with mean zero with standard deviations of $\sigma_e = 1/\sqrt{2}$ and $\sigma = \sigma_\zeta/\sqrt{2}$ with $\sigma_{\sigma_\zeta} = 10^{-7}$ respectively. The number of decoding trials is set to be $N = \tau_g - 1$, the initial number of erasures $\varepsilon = 0$ and the increase in erasures per trial $\varepsilon^+ = 1$.

Each algorithm considered in this simulation provides the output $\widehat{\boldsymbol{e}}$, which is the estimation of the original sparse vector $\boldsymbol{e}$. The level of performance is determined by observing the squared Euclidean distance $(d_{\mathrm{E}}(\boldsymbol{e}, \widehat{\boldsymbol{e}}))^2 = d_{\mathrm{E}}^2(\boldsymbol{e}, \widehat{\boldsymbol{e}})$. The distribution of the squared Euclidean distance $d_{\mathrm{E}}^2(\boldsymbol{e}, \widehat{\boldsymbol{e}})$ is plotted using *boxplots*. There exist several different variants of boxplots [FHI89]. Similar to the boxplots used in Section 5.4.1, we use the model provided by Tukey [Tuk77]. For reading convenience, we revisit the most important properties shown in a boxplot: The main part of the boxplot is built by a rectangle, which resembles the values between first and third quartile. The median is represented by a black horizontal bar within this box and the mean (average) is shown as a circle.

For the $\mathcal{CRS}(32, 8)$ code, the decoding radii of different decoding methods used are as follows: GS-based GMD decoding radius is $\tau_g = 15$, half minimum distance (BMA) $\tau = 12$ and power decoding radius $\tau_p = 13$. The results for the simulation for the noiseless and noisy cases are shown in Figures 5.9 and 5.10 respectively. In the noiseless scenario, Figure 5.9 shows that the newly proposed algorithm provides satisfactory decoding results up to its radius, thus outperforming its competitors in terms of number of errors. It is even able to find a highly accurate estimation of the sparse error vector, which is almost as good as the other two algorithms. However, it should be noted that a few insignificant outliers exist for some of results. The number of outliers increases considerably when noise is added. Figure 5.10, representing the noisy scenario, shows that even when the new algorithm is able to correct more errors, it loses its accuracy for an exceptionally low noise level ($\sigma_\zeta = 10^{-7}$). This should have been expected, especially after showing that interpolation-based decoding is extremely sensitive to noise and numerical inaccuracies, which would obstruct the chances of getting a flawless and accurate result,.

The same behaviour is observed when using a shorter $\mathcal{CRS}(16, 4)$ code. The new decoding radii are as follows: GS-based GMD decoding radius is $\tau_g = 8$, half minimum distance $\tau = 6$ and power decoding radius $\tau_p = 7$. The noiseless and noisy scenarios are shown in Figures 5.11 and 5.12 respectively. In Figure

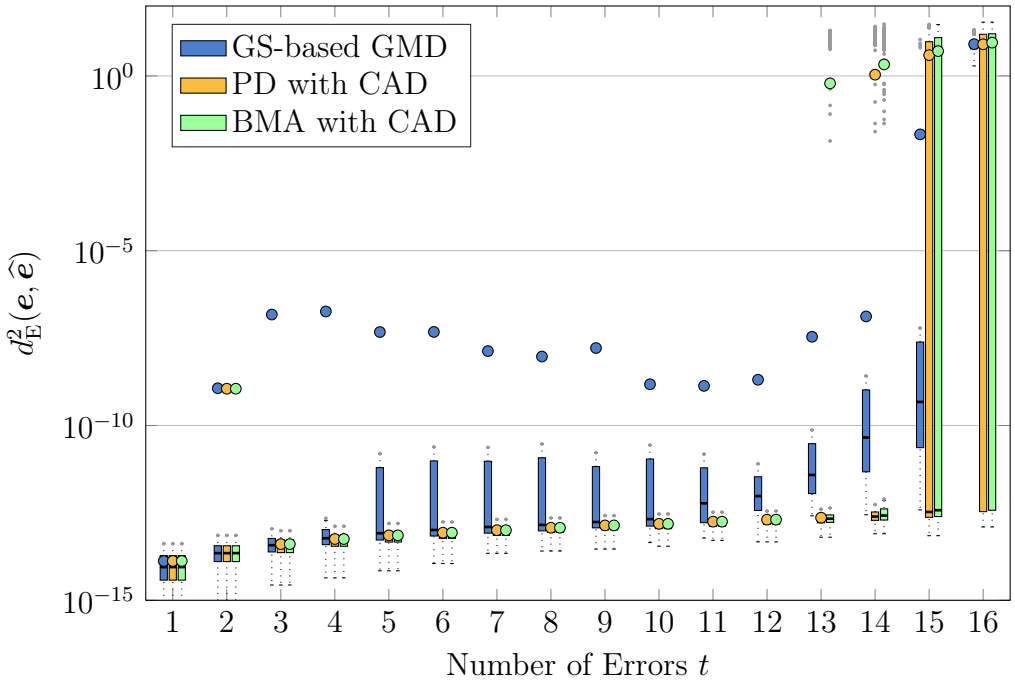

Figure 5.10: Boxplots illustrating the distribution of the squared Euclidean distance $d_{\mathrm{E}}^2(\boldsymbol{e}, \widehat{\boldsymbol{e}})$ in a noisy scenario ($\sigma_{\boldsymbol{\varsigma}} = 10^{-7}$) for a $\mathcal{CRS}(32, 8)$ code for different decoding algorithms.

5.12, the effect of noise can still be seen, although its impact is not as large as for the longer code. This means the length of the code plays a role in the decoding performance in a noisy scenario. However, it should not be forgotten that the degradation in performance comes from numerical inaccuracies arising from the root finding step, which increases exponentially with the dimension k.

From these results, we show that it is possible to use the interpolation-based decoding for sparse reconstruction in a deterministic CS based on CRS codes scheme. Overcoming existing obstacles, such as inaccurate results produced by the root-finding algorithm can be done by a few modifications and the use of Newton's method. Numerical simulations show that the proposed algorithm is able to function properly while having a good performance and an increased decoding radius when compared to syndrome-based decoding algorithms. However, in a noisy scenario, syndrome-based decoding shows more robustness to numerical instability and noise. The proposed algorithm is extremely sensitive to noise, which is still a concern to be dealt with. Studying the effect of the choice of parameters as well as of other cost efficient root finding algorithms on

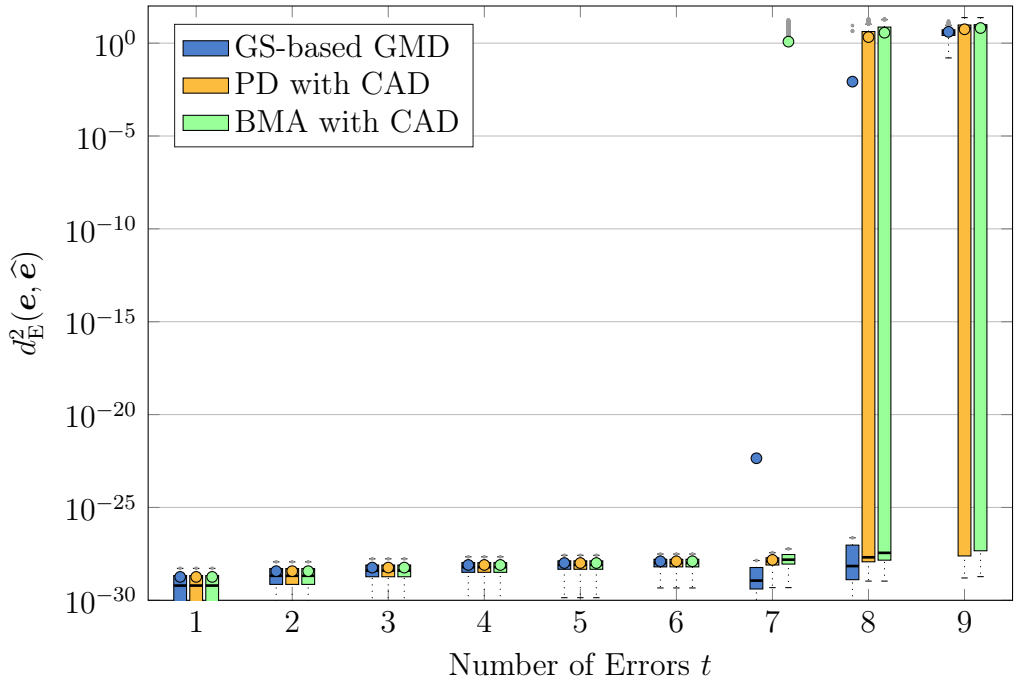

Figure 5.11: Boxplots illustrating the distribution of the squared Euclidean distance $d_{\mathrm{E}}^{2}(\boldsymbol{e}, \widehat{\boldsymbol{e}})$ in a noiseless scenario for a $\mathcal{CRS}(16, 4)$ code for different decoding algorithms [MPB17].

the performance and stability should be one of the points to be addressed in any future research in the topic.

5.5 Overview and Summary

This chapter focuses on the use of CRS codes in the application of deterministic CS. Known algebraic decoding algorithms can be used in sparse vector reconstruction. It is shown in [Zö15] that when using a classical decoders (like BMA or EEA) beyond their capabilities for sparse reconstruction, it is possible to obtain reliability information for further decoding attempts.

With the help of this reliability information, we are in a position to use the concept of error/erasure decoding to achieve a better performance when compared to that of the known CS sparse reconstruction schemes like OMP. Another possible use of the reliability information is overcoming the numerical inaccuracies that occur in interpolation-based decoding algorithms when used

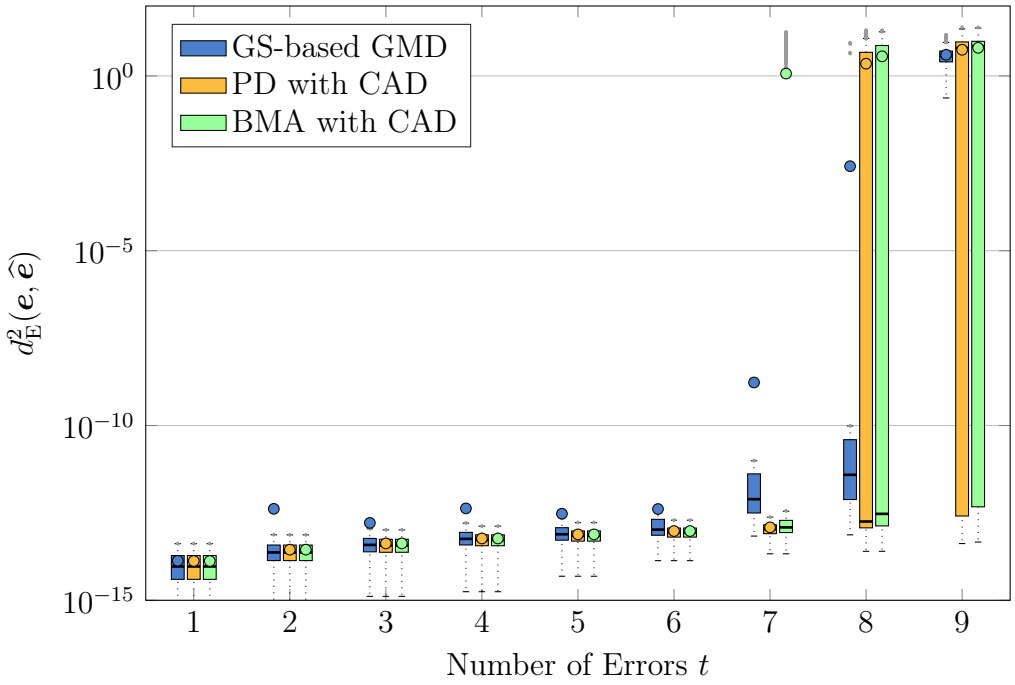

Figure 5.12: Boxplots illustrating the distribution of the squared Euclidean distance $d_{\mathrm{E}}^{2}(\boldsymbol{e}, \widehat{\boldsymbol{e}})$ in a noisy scenario ($\sigma_{\varsigma} = 10^{-7}$) for a $\mathcal{CRS}(16, 4)$ code for different decoding algorithms [MPB17].

for applications in the complex field.

First, we explain the origin of the reliability information obtained from decoding CRS codes. We also show different methods of calculating it and offer a comparison between the quality of the reliability information they provide. In the process of studying these methods, we come across some of the properties they attain, namely the robustness against noise and the rate dependence.

After that, we discuss two algorithms which make use of the calculated reliability information. The first investigated algorithm is the REA [MZB16]. With the use of error/erasure decoding, the multi-trial algorithm produces an enhanced version of the reliability information, allowing an increase in performance when compared to pre-enhancement results. It even outperform the OMP algorithm even when coupled with an optimized matrix to achieve its best results.

We also investigate the use of interpolation-based algorithms for the application of deterministic CS. Algorithms such as the GS algorithm were always avoided when dealing with complex numbers. That is because of the sensitivity of the root finding phase to numerical inaccuracies. Through a few modifica-

tions and a few added steps like the use of Newton's method, it was established in [MZB16] that the GS algorithm can still function properly even when considering an application over the complex field. For the use in sparse reconstruction, the algorithms used should output a single solution. The GS algorithm, however, is considered a list decoder; producing a list of possible solutions as its output. Therefore, an algorithm which utilizes reliability information, the GS-based GMD decoder, is introduced in order to achieve this requirement. By numerical simulations, it is shown that it provides a better performance when compared to other syndrome-based decoders like CAD. It is still however more vulnerable to numerical inaccuracies, especially when affected by noise.

Although CRS codes are over 30 years old, we consider investigating their use in various applications a young field of research. It was not long ago that they were discovered to be used for deterministic CS. Even if the algorithms investigated in this work, both REA and GS-based GMD, deliver some of the best known results, they are far from being perfect. The effect of changing the parameters on their performance should be further studied and optimized. The functionality of many other RS decoding algorithms, like Wu or KV, in the complex field is still an open question.

6

Conclusion

Within this dissertation, algebraic decoding of Reed–Solomon (RS) codes, defined over finite fields as well as the complex field, is investigated. With the aid of reliability information, it is possible to reach better performance by decoding beyond half the minimum distance. In some cases, its is even possible to reduce the complexity of some known decoders.

In the course of this thesis, four decoding algorithms were developed specifically for these purposes: Two algorithms for RS codes defined over finite fields and another two for RS codes defined over the complex field. Each pair consist of a syndrome-based decoder and an interpolation-based decoder. All these new algorithms are inspired by the the well-known Generalized Minimum Distance (GMD) decoding technique [For66], which is a multi-trial technique where error/erasure decoding is performed with the number of erasures is varied in each trial. Classical RS decoders, like the Berlekamp–Massey algorithm, are used as building blocks for the new algorithms and a brief explanation of them can be found in Chapter 3. The main results can be summerized in the following.

For RS codes defined over finite fields, the Chase-like and the reduced-list decoders are explained and their performance evaluated in Chapter 4. The Chase-like decoder, named after the original Chase decoder, utilizes reliability information to perform multiple error/erasure decoding attempts without repeating the whole decoding process from scratch. It is able to outperform the Kötter–Vardy decoder, provided they both have the same limited complexity. The Chase-like decoder is most suitable for decoding high-rate codes, where correcting one or two more errors beyond half the minimum distance is considered valuable. To correct more than two errors, the computational cost becomes exponentially large and other decoding techniques are recommended. A more cost efficient algorithm can be seen in the reduced-list decoder, which is developed from the Wu algorithm. Using the reliability information, we introduce a smaller

set of interpolation points consisting of the least reliable positions. Initially the goal was to reduce the complexity of the decoding process, however, it turns out that this reduction enhances the error correction capabilities. The performance of our low complexity decoder surpasses that of a Kötter–Vardy decoder having a huge computational cost. Although, its performance is considered more than acceptable, parameters like the reduced number of interpolation points is yet to be optimized and should be subjected to further analysis.

It was recently discovered that RS codes defined over the complex field can be used in Compressed Sensing (CS). Decoding algorithms can be viewed as sparse vector reconstruction algorithms. Even more recently, it was shown that reliability information can be extracted from a failed decoding attempt. In Chapter 5, we show new methods of extracting reliability information and investigate their properties. We also develop the Recursive Enhancement Algorithm (REA) and the Guruswami–Sudan (GS)-based GMD decoder. The REA focuses on enhancing the quality of the available reliability information through multiple error/erasure decoding trials. The enhanced reliability information is then input to the continuity assisted decoder from [Zö15]. With the provided enhancement, the decoder exceeds its previous results. It also surpasses the performance of Orthogonal Matching Pursuit, which is a popular sparse reconstruction scheme. The number of iterations and the number of erasures introduced in an iteration are not yet optimized. A more detailed study on their influence can be considered as an invited question for future researchers.

Due to the presence of numerical inaccuracies when dealing with complex numbers, interpolation-based decoders are usually avoided due to their high sensitivity. In an attempt to show that interpolation-based algorithms, like GS, can still be used for sparse reconstruction, the GS-based GMD decoder was established. To counter the effect of the numerical inaccuracies, modifications such as altering the Roth-Ruckenstein root finding algorithm and using Newton's method are proposed. In order to provide a single solution for the sparse reconstruction instead of a list of possible solutions, reliability information is used in a GS-based GMD decoder. The overall decoder provides a more stable performance with a correction radius exceeding the classical half minimum distance decoding. However, results show that syndrome-based decoders are still more robust to numerical inaccuracies and noise. Therefore, it is recommended to use the new interpolation-based algorithm only in a noise free scenario.

Bibliography

References

[ADAA08] F. Abdelkefi, P. Duhamel, F. Alberge, and J. Ayadi. On the use of cascade structure to correct impulsive noise in multicarrier systems. *IEEE Trans. Comm.*, 56(11):1844–1858, November 2008. doi:10.1109/TCOMM.2008.060532. Cited on page 53.

[Ale05] M. Alekhnovich. Linear Diophantine Equations over Polynomials and Soft Decoding of Reed-Solomon Codes. *IEEE Trans. Inf. Theory*, 51(7):2257–2265, July 2005. Cited on page 75.

[AMM12] A. Amini, V. Montazerhodjat, and F. Marvasti. Matrices with small coherence using p-ary block codes. *IEEE Trans. Signal Process.*, 60(1):172–181, January 2012. doi:10.1109/TSP.2011.2169249. Cited on page 57.

[AT08] M. Akcakaya and Vahid Tarokh. A frame construction and a universal distortion bound for sparse representations. *IEEE Trans. Sign. Proces.*, 56(6):2443–2450, June 2008. doi:10.1109/TSP.2007.914344. Cited on page 53.

[BB08] M. Bossert and S. Bezzateev. Decoding of Interleaved RS Codes with the Euclidean Algorithm. In *IEEE International Symposium on Information Theory, 2008. ISIT 2008.*, pages 1803–1807, July 2008. doi:10.1109/ISIT.2008.4595299. Cited on pages 25 and 37.

[BB10] P. Beelen and K. Brander. Key Equations for List Decoding of Reed–Solomon Codes and How to Solve Them. *J. Symbolic Comp.*, 45(7):773–786, 2010. Cited on page 75.

[BB13] M. Bossert and S. Bezzateev. A unified view on known algebraic decoding algorithms and new decoding concepts. *IEEE Transactions on Information Theory*, 59(11):7320–7336, Nov 2013. doi:10.1109/TIT.2013.2274454. Cited on page 16.

[Ber68] E. R. Berlekamp. *Algebraic coding theory.* McGraw–Hill, 1968. Cited on pages 2, 17, 19, and 20.

[BGM96] George A. Baker and Peter R. Graves-Morris. *Padé approximants.* Cambridge University Press, Cambridge, 2. ed. edition, 1996. Cited on page 58.

[Bos99] Martin Bossert. *Channel Coding for Telecommunications.* Wiley, 1999. Cited on pages 6, 7, 9, 10, 17, 19, 30, and 34.

[BRC60] R.C. Bose and D.K. Ray-Chaudhuri. On a Class of Error Correcting Binary Group Codes. *Information and Control*, 3(1):68 – 79, 1960. doi:http://dx.doi.org/10.1016/S0019-9958(60)90287-4. Cited on pages 1 and 5.

[CBL89] S. Chen, S. A. Billings, and W. Luo. Orthogonal least squares methods and their application to non-linear system identification. *Int. J. Control*, 50:1873–1896, 1989. Cited on pages 3, 56, and 71.

[Cha72] D. Chase. Class of algorithms for decoding block codes with channel measurement information. *IEEE Transactions on Information Theory*, 18(1):170–182, Jan 1972. doi:10.1109/TIT.1972.1054746. Cited on pages 2, 3, 33, and 37.

[CHJ10] R. Calderbank, S. Howard, and Sina Jafarpour. Construction of a large class of deterministic sensing matrices that satisfy a statistical isometry property. *IEEE J. Sel. Top. Sign. Proces.*, 4(2):358–374, April 2010. doi:10.1109/JSTSP.2010.2043161. Cited on page 57.

[CS03] Don Coppersmith and Madhu Sudan. Reconstructing curves in three (and higher) dimensional space from noisy data. In *Proc. 35th Symp. Theory Comput.*, page 136–142, New York, USA, 2003. ACM. doi:10.1145/780542.780563. Cited on pages 54 and 57.

[CSS14] Anas Chaaban, Vladimir Sidorenko, and Christian Senger. On multitrial forney-kovalev decoding of concatenated codes. *Advances in Mathematics of Communications*, 8(1):1–20, 2014. doi:10.3934/amc.2014.8.1. Cited on page 34.

[CT06] E. J. Candes and T. Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Transactions on*

Information Theory, 52(12):5406–5425, Dec 2006. doi:10.1109/TIT.2006.885507. Cited on page 56.

[DE03] David L. Donoho and Michael Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization. *Proc. Natl. Acad. Sci. USA*, 100(5):2197–2202, 2003. doi:10.1073/pnas.0437847100. Cited on page 56.

[Del75] P. Delsarte. On subfield subcodes of modified reed-solomon codes (corresp.). *IEEE Trans. Inf. Theor.*, 21(5):575–576, September 1975. doi:10.1109/TIT.1975.1055435. Cited on page 10.

[DeV07] Ronald A. DeVore. Deterministic constructions of compressed sensing matrices. *J. Complexity*, 23(4-6):918 – 925, 2007. doi:DOI:10.1016/j.jco.2007.04.002. Cited on page 57.

[Don06] David L. Donoho. Compressed sensing. *IEEE Trans. Information Theory*, 52(4):1289–1306, 2006. doi:10.1109/TIT.2006.871582. Cited on page 56.

[Dor74] B. Dorsch. A decoding algorithm for binary block codes and j -ary output channels (corresp.). *IEEE Transactions on Information Theory*, 20(3):391–394, May 1974. doi:10.1109/TIT.1974.1055217. Cited on pages 2 and 33.

[DR08] Wolfgang Dahmen and Arnold Reusken. *Numerik für Ingenieure und Naturwissenschaftler*. Springer, Berlin; Heidelberg, 2., korr. aufl. edition, 2008. Cited on pages 75 and 77.

[EK12] Yonina C. Eldar and Gitta Kutyniok. *Compressed Sensing: Theory and Applications*. Cambridge UP, 2012. Cited on pages 7 and 56.

[FHI89] Michael Frigge, David C. Hoaglin, and Boris Iglewicz. Some implementations of the box plot. *The American Statistician*, 43(1):50–54, February 1989. Cited on pages 71 and 82.

[Fit95] Patrick Fitzpatrick. On the key equation. *IEEE Trans. Inf. Theory*, 41(5):1290–1302, 1995. Cited on page 59.

[For65] G. D. Forney. On decoding BCH codes. *IEEE Trans. Inf. Theory*, 11:549–557, October 1965. Cited on page 22.

[For66] G. Forney. Generalized Minimum Distance Decoding. *IEEE Transactions on Information Theory*, 12(2):125–131, April 1966. doi:10.1109/TIT.1966.1053873. Cited on pages 2, 29, 31, 33, 34, and 87.

[FR13] Simon Foucart and Holger Rauhut. *A mathematical introduction to compressive sensing*. Appl. Numer. Harmon. Anal. Birkhäuser, 2013. Cited on page 56.

[GS99] V. Guruswami and M. Sudan. Improved Decoding of Reed-Solomon and Algebraic-Geometry Codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, Sep. 1999. doi:10.1109/18.782097. Cited on pages 2, 23, and 24.

[GVL96] Gene H. Golub and Charles F. Van Loan. *Matrix computations*. Johns Hopkins Univ. Pr., Baltimore, Md., 3. ed. edition, 1996. Cited on page 75.

[GZ61] D. C. Gorenstein and N. Zierler. A class of error-correcting codes in p^m. *J. Soc. Indust. Appl. Math.*, 9:207–214, June 1961. Cited on page 22.

[Ham50] R. W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29(2):147–160, 1950. doi:10.1002/j.1538-7305.1950.tb00463.x. Cited on pages 1, 5, and 7.

[Hen89] W. Henkel. *Zur Decodierung algebraischer Blockcodes über komplexen Alphabeten*. Fortschritt-Berichte VDI. VDI-Verlag, 1989. Cited on page 53.

[Hen00] Werner Henkel. Analog codes for peak-to-average ratio reduction. In *Proc. 3rd ITG Conf. Source Channel Coding*, Munich, January 2000. Cited on page 53.

[HH05] Werner Henkel and Fang Ning Hu. OFDM and analog RS/BCH codes. In *Proc. OFDM-Workshop*, Hamburg, August 2005. Cited on page 53.

[HHH12] M. Huemer, C. Hofbauer, and J. B. Huber. Non-systematic complex number RS coded OFDM by unique word prefix. *IEEE Trans. Signal Process.*, 60(1):285–299, January 2012. Cited on page 53.

[HHZ11] Fang Ning Hu, Werner Henkel, and Ming Jie Zhao. Analog codes for gross error correction in signal transmission. *Adv. Mat. Res.*, 341–342:514–518, September 2011. doi:10.4028/www.scientific.net/AMR.341-342.514. Cited on page 53.

[ITU99] International Telecommunication Union ITU. Asymmetric Digital Subscreiber line (ADSL) transceivers. G.992.1, July 1999. Cited on page 11.

[KNIH94] T. Kaneko, T. Nishijima, H. Inazumi, and S. Hirasawa. An Efficient Maximum-Likelihood-Decoding Algorithm for Linear Block Codes With Algebraic Decoder. *IEEE Transactions on Information Theory*, 40(2):320–327, March 1994. Cited on page 35.

[Kum85] R. Kumaresan. Rank reduction techniques and burst error-correction decoding in real/complex fields. In *Proc. Asilomar Conf. Sign., Syst. Comp.*, pages 457–461, November 1985. doi:10.1109/ACSSC.1985.671503. Cited on page 53.

[KV03] R. Kötter and A. Vardy. Algebraic Soft-Decision Decoding of Reed-Solomon Codes. *IEEE Transactions on Information Theory*, 49(11):2809–2825, Nov. 2003. doi:10.1109/TIT.2003.819332. Cited on pages 2, 24, 33, and 42.

[KWB10] S. Kampf, A. Wachter, and M. Bossert. A method for soft-decision decoding of reed-solomon codes based on the extended euclidean algorithm. In *International ITG Conference on Source and Channel Coding (SCC)*, pages 1–6, Jan. 2010. Cited on page 35.

[LN02] Rudolf Lidl and Harald Niederreiter. *Introduction to finite fields and their applications*. Cambridge Univ. Press, Cambridge [u.a.], rev. ed. edition, 2002. Cited on page 5.

[Mar84] T. Jr. Marshall. Coding of real-number sequences for error correction: A digital signal processing problem. *IEEE J. Select. Areas Commun.*, 2(2):381–392, March 1984. doi:10.1109/JSAC.1984.1146063. Cited on pages 53 and 68.

[MHET99] F. Marvasti, M. Hasan, M. Echhart, and S. Talebi. Efficient algorithms for burst error recovery using FFT and other transform kernels. *IEEE Trans. Sign. Proces.*, 47(4):1065–1075, April 1999. doi:10.1109/78.752604. Cited on page 53.

[MS85] Y. Maekawa and K. Sakaniwa. An extension of DFT code and the evaluation of its performance. In *Proc. IEEE Int. Symp. Inf. Theory*, pages 24–28, June 1985. Cited on page 53.

[MS88] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North Holland Publishing Co., June 1988. Cited on pages 2, 8, 9, 10, 17, and 33.

[MT11] V. Miloslavskaya and P. Trifonov. Hybrid interpolation Algorithm for Algebraic Soft Decision Decoding of Reed-Solomon Codes. In *8th International Symposium on Wireless Communication Systems (ISWCS)*, pages 131–135, Nov. 2011. doi:10.1109/ISWCS.2011.6125324. Cited on page 41.

[Nat95] B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM J. Comput.*, 24(2):227–234, April 1995. doi:10.1137/S0097539792240406. Cited on page 56.

[Nie13] Johan S. R. Nielsen. *List Decoding of Algebraic Codes*. PhD thesis, Technical University of Denmark, 2013. Cited on pages 26 and 27.

[Nie16] Johan S. R. Nielsen. Power Decoding Reed–Solomon Codes up to The Johnson Radius. *Advances in Mathematics of Communications (AMC)*, 10, November 2016. Cited on pages 21 and 61.

[Pad92] H. Padé. Sur la représentation approchée d'une fonction par des fractions rationnelles. *Annales scientifiques de l'École Normale Supérieure*, 9:3–93, 1892. Cited on page 58.

[PH08] F. Parvaresh and B. Hassibi. Explicit measurements with almost optimal thresholds for compressed sensing. In *IEEE Int. Conf. Acoust., Speech, Signal Process.*, pages 3853–3856, March 2008. doi:10.1109/ICASSP.2008.4518494. Cited on pages 54, 57, and 75.

[PW72] W. W. Peterson and E. J. Weldon. *Error-Correcting Codes*. MIT Press, second edition, 1972. Cited on pages 2 and 59.

[Red00] G.R. Redinbo. Decoding real block codes: activity detection wiener estimation. *IEEE Trans. Inf. Theory*, 46(2):609–623, March 2000. doi:10.1109/18.825828. Cited on page 53.

[RG03] G. Rath and C. Guillemot. Real error and erasure correction with dft codes for communication channels. In *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003.*, volume 1, pages 311–316 vol.1, March 2003. doi:10.1109/WCNC.2003.1200366. Cited on page 53.

[RHG01] G. Rath, X. Henocq, and C. Guillemot. Application of dft codes for robustness to erasures. In *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, volume 2, pages 1246–1250 vol.2, 2001. doi:10.1109/GLOCOM.2001.965685. Cited on page 53.

[Rot06] Ron M. Roth. *Introduction to coding theory.* Cambridge UP, 2006. Cited on pages 7, 10, 27, 28, 30, and 34.

[RR00] R. M. Roth and G. Ruckenstein. Efficient decoding of reed-solomon codes beyond half the minimum distance. *IEEE Transactions on Information Theory*, 46(1):246–257, Jan 2000. doi:10.1109/18.817522. Cited on pages 23, 27, and 28.

[RS60] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960. Cited on pages 1, 5, and 10.

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27:379–423,623–656, 1948. Cited on page 1.

[SSB06] G. Schmidt, V. Sidorenko, and M. Bossert. Decoding Reed–Solomon codes beyond half the minimum distance using shift-register synthesis. In *Proc. IEEE Int. Symp. Inf. Theory*, pages 459–463, July 2006. doi:10.1109/ISIT.2006.261711. Cited on pages 12 and 21.

[SSB10] G. Schmidt, V. R. Sidorenko, and M. Bossert. Syndrome decoding of Reed–Solomon codes beyond half the minimum distance based on shift-register synthesis. *IEEE Trans. Inf. Theory*, 56:5245–5252, October 2010. Cited on pages 17, 20, 22, and 61.

[SSBZ10] C. Senger, V. R. Sidorenko, M. Bossert, and V. V. Zyablov. Multitrial decoding of concatenated codes using fixed thresholds. *Problems of Information Transmission*, 46(2):127–141, 2010. doi:10.1134/S0032946010020031. Cited on page 34.

[Sud97] Madhu Sudan. Decoding of reed solomon codes beyond the error-correction bound. *J. Complex.*, 13(1):180–193, March 1997. doi:10.1006/jcom.1997.0439. Cited on pages 2, 23, and 24.

[TH08] G. Takos and C.N. Hadjicostis. Determination of the number of errors in DFT codes subject to low-level quantization noise. *IEEE Trans. Sign. Proces.*, 56(3):1043–1054, March 2008. doi:10.1109/TSP.2007.908939. Cited on page 53.

[Tri07] P. Trifonov. Interpolation in List Decoding of Reed–Solomon Codes. *Probl. Inf. Transm.*, 43(3):190–198, 2007. Cited on page 75.

[Tri10] P. Trifonov. Another Derivation of Wu List Decoding Algorithm and Interpolation in Rational Curve Fitting. In *IEEE SIBIRCON*, pages 59–64, July 2010. doi:10.1109/SIBIRCON.2010.5555314. Cited on pages 26 and 48.

[Tuk77] John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977. Cited on pages 71 and 82.

[VL14] M. Vaezi and F. Labeau. Generalized and extended subspace algorithms for error correction with quantized DFT codes. *IEEE Trans. Comm.*, 62(2):410–422, February 2014. doi:10.1109/TCOMM.2014.010414.130440. Cited on page 53.

[WB86] L.R. Welch and E.R. Berlekamp. Error correction for algebraic block codes, December 30 1986. US Patent 4,633,470. URL: https://www.google.com/patents/US4633470. Cited on pages 2, 23, and 24.

[WB94] S.B. Wicker and V.K. Bhargava. *Reed-Solomon Codes and Their Applications*. IEEE Press, 1994. Cited on pages 1, 10, and 11.

[Wol83] J. K. Wolf. Redundancy, the discrete Fourier transform, and impulse noise cancellation. *IEEE Trans. Comm.*, 31(3):458–461, March 1983. doi:10.1109/TCOM.1983.1095820. Cited on pages 53, 55, and 68.

[Wu08] Yingquan Wu. New List Decoding Algorithms for Reed–Solomon and BCH Codes. *IEEE Transactions on Information Theory*, 54(8):3611–3630, July 2008. doi:10.1109/TIT.2008.926355. Cited on pages 2, 3, 23, 25, and 27.

[ZB15] Henning Zörlein and Martin Bossert. Coherence optimization and best complex antipodal spherical codes. *IEEE Trans. Signal Process.*, 63(24):6606–6615, December 2015. doi:10.1109/TSP.2015.2477052. Cited on pages 56 and 71.

[ZGA11] Alexander Zeh, Christian Gentner, and Daniel Augot. An Interpolation Procedure for List Decoding Reed–Solomon Codes Based on Generalized Key Equations. *IEEE Trans. Inf. Theory*, 57(9):5946–5959, 2011. Cited on page 75.

[Zö15] Henning Zörlein. *Channel Coding Inspired Contributions to Compressed Sensing*. PhD thesis, Ulm University, 2015. URL: http://vts.uni-ulm.de/doc.asp?id=9748. Cited on pages 3, 54, 56, 57, 58, 60, 61, 62, 64, 68, 69, 71, 81, 84, and 88.

Publications Containing Parts of this Thesis

[MNB14] Mostafa H. Mohamed, Johan S. R. Nielsen, and Martin Bossert. Reduced list-decoding of Reed–Solomon codes using reliability information. In *Proceedings of the 21st International Symposium on Mathematical Th. of Networks and Systems*, Jul 2014. Cited on pages 44 and 52.

[MB15] Mostafa H. Mohamed, and Martin Bossert. A Chase-like decoding algorithm for Reed–Solomon codes based on the extended Euclidean algorithm. In *Proc. Int. ITG Conf. Sys., Commun. Coding*, Feb 2015. Cited on pages 3, 37, and 52.

[MRZB15] Mostafa H. Mohamed, Shrief Rizkalla, Henning Zörlein, and Martin Bossert. Deterministic compressed sensing with power decoding for complex Reed–Solomon codes. In *Proc. Int. ITG Conf. Sys., Commun. Coding*, Feb 2015. Cited on pages 54, 56, 57, and 68.

[MZB16] Mostafa H. Mohamed, Henning Zörlein, and Martin Bossert. Recursive enhancement of intrinsic soft information for complex Reed–Solomon codes. In *4th Int. Workshop on Compressed Sensing Theory and its Applications to Radar, Sonar and Remote Sensing*, Sep 2016. Cited on pages 3, 61, 68, 70, 85, and 86.

[MPB17] Mostafa H. Mohamed, Sven Puchinger, and Martin Bossert. Guruswami–Sudan list decoding for complex Reed–Solomon codes. In

Proc. Int. ITG Conf. Sys., Commun. Coding, Feb 2017. Cited on pages 3, 57, 75, 76, and 80.

Mostafa Hosni Mohamed

Curriculum Vitae

For reasons of data privacy, the curriculum vitae has been removed.

List of Publications

[1] **M. H. Mohamed**, Johan S. R. Nielsen, and Martin Bossert. Reduced List-Decoding of Reed–Solomon Codes Using Reliability Information. In *Proceedings of the 21st International Symposium on Mathematical Theory of Networks and Systems (MTNS)*, Groningen, the Netherlands, July 2014.

[2] **M. H. Mohamed** and Martin Bossert. A Chase-like Decoding Algorithm for Reed–Solomon Codes Based on the Extended Euclidean Algorithm. In *Proceedings of the 10th International ITG Conference on Systems, Communications and Coding (SCC)*, Hamburg, Germany, February 2015

[3] Henning Zörlein, Shrief Rizkalla, **M. H. Mohamed**, and Martin Bossert. Deterministic Compressed Sensing with Power Decoding for Complex Reed-Solomon Codes. In *Proceedings of the 10th International ITG Conference on Systems, Communications and Coding (SCC)*, Hamburg, Germany, February 2015.

[4] **M. H. Mohamed** and Martin Bossert. Combinatorial Metrics and Collaborative Error/Erasure Decoding for Translational Metrics. In *Proceedings of the IITP RAS 39th interdisciplinary School-Conference Information Technologies and Systems (ITaS)*, Sochi, Russia, September 2015.

[5] **M. H. Mohamed**, Henning Zörlein, and Martin Bossert. Recursive Enhancement of Intrinsic Soft Information for Complex Reed-Solomon Codes. In *Compressed Sensing Theory and its Applications to Radar, Sonar and Remote Sensing (CoSeRa)*, Aachen, Germany, September 2016.

[6] **M. H. Mohamed**, Sven Puchinger, and Martin Bossert. Guruswami–Sudan List Decoding for Complex Reed–Solomon Codes. In *Proceedings of the 11th International ITG Conference on Systems, Communications and Coding (SCC)*, Hamburg, Germany, February 2017.