Reed–Solomon Codes over Fields of Characteristic Zero

Carmen Sippel\textsuperscript{1}, Cornelia Ott\textsuperscript{1}, Sven Puchinger\textsuperscript{2}, and Martin Bossert\textsuperscript{1}

\textsuperscript{1}Institute of Communications Engineering, Ulm University, Germany
\textsuperscript{2}Institute for Communications Engineering, Technical University of Munich (TUM), Germany

\{carmen.sippel, cornelia.ott, martin.bossert\}@uni-ulm.de, sven.puchinger@tum.de

Abstract—We study Reed–Solomon codes over arbitrary fields, inspired by several recent papers dealing with Gabidulin codes over fields of characteristic zero. Over the field of rational numbers, we derive bounds on the coefficient growth during encoding and the bit complexity of decoding, which is polynomial in the code length and in the bitsize of the error values. The results can be generalized to arbitrary number fields.

I. INTRODUCTION

Reed–Solomon (RS) codes were introduced in 1960 [1] and have become some of the most used classes of algebraic codes. The codes are defined over finite fields and can be efficiently decoded, both up to half-the-minimum distance [2], [3] and beyond [4], [5], [6], [7], [8].

RS codes over the field of complex numbers, also called complex RS codes or analog codes, were introduced in [9] and [10], and their decoding was studied in [11]. Further decoding principles known from finite fields, as well as applications of the codes to compressed sensing, were analyzed and numerically evaluated in [12], [13], [14], [15].

In this paper, we study RS codes over arbitrary fields and their properties. The idea is inspired by several recent publications that have dealt with Gabidulin codes, the rank-metric analog of Reed–Solomon codes, over field of characteristic zero (in particular number fields), which were first described in [16, Section 6], [17], [18], [19], [20]. These codes have applications in space-time coding [21] and low-rank matrix recovery [22]. Among other possible applications, we believe that RS codes over number fields (in particular over Q and Q[i]) are suitable to replace complex RS codes in some applications, e.g., the compressed sensing scenario studied in [13], [14], since there are no numerical issues. Studying these applications goes beyond the scope of this paper and needs to be done in future work.

For decoding, we adapt a syndrome-based half-the-minimum-distance decoder known from finite fields and restrict to the field of rational numbers, which is an infinite field and an exact computation domain. In contrast to complex RS codes, we face the problem of coefficient growth (i.e., large numbers, we derive bounds on the coefficient growth during encoding and decoding. This implies an upper bound on the bit complexity of decoding, which is polynomial in the code length and bitsize of the error values.

The results can be extended to more classes of number fields, for instance Q[i], or cyclotomic and Kummer extensions. The adaptation is technical, which is why we restrict ourselves to Q here. Furthermore, it is reasonable to expect that our results can be used to analyze the runtimes of the existing decoders for Gabidulin codes over number fields [17], [18], [19] in bit operations, instead of field operations (which do not consider coefficient growth).

II. REED–SOLONOM CODES OVER ARBITRARY FIELDS

A. Definition

The following is a straightforward generalization of generalized Reed–Solomon (GRS) codes to arbitrary, possibly infinite, fields. We define the codes as in [3] by simply replacing the finite field by an arbitrary field K.

Definition 1. Let k and n be integers such that k < n. Furthermore, choose $\alpha_1, \ldots, \alpha_n \in K$ to be distinct non-zero elements of the field K, and $v_1, \ldots, v_n$ to be non-zero elements of K. The corresponding generalized Reed–Solomon (GRS) code is defined by the linear code $C_{GRS} \subseteq K^n$ with parity check matrix

$$
H_{GRS} = \begin{pmatrix}
1 & 1 & \ldots & 1 \\
\alpha_1 & \alpha_2 & \ldots & \alpha_n \\
\vdots & \vdots & \ddots & \vdots \\
\alpha_1^{n-k-1} & \alpha_2^{n-k-1} & \ldots & \alpha_n^{n-k-1}
\end{pmatrix}
\begin{pmatrix}
v_1 \\
v_2 \\
\vdot
\vdot
\vdot \\
v_n
\end{pmatrix}.
$$

The proof that RS codes are MDS is straightforward using the same arguments as in the finite field case (see, e.g., [3]).

Theorem 1. Any GRS code is MDS, i.e., d = n − k + 1.

Theorem 2. The code $C_{GRS}$ as defined in Definition 1 has a generator matrix of the form

$$
G_{GRS} = \begin{pmatrix}
1 & 1 & \ldots & 1 \\
\alpha_1 & \alpha_2 & \ldots & \alpha_n \\
\vdots & \vdots & \ddots & \vdots \\
\alpha_{k-1} & \alpha_{k-1} & \ldots & \alpha_{n-1}
\end{pmatrix}
\begin{pmatrix}
v'_1 \\
v'_2 \\
\vdot
\vdot
\vdot \\
v'_n
\end{pmatrix},
$$

where the $\alpha_i$ are the same as in Definition 1 and the $v'_i$ are non-zero elements of K, given by the following linear system of equations:

$$
\sum_{j=1}^{n} \alpha_i^r v'_j = 0 \quad \forall r = 0, \ldots, n-2.
$$

Proof: The proof is analogous to [3, Proposition 5.2].

Theorem 3. Fix $\alpha_1, \ldots, \alpha_n \in K \setminus \{0\}$ distinct. Let $v_i, v'_i$ be non-zero such that

$$
v_i v'_j = \prod_{1 \leq j \leq n, i \neq j} (\alpha_i - \alpha_j)^{-1} =: w_i.
$$

Then, the corresponding matrices $G_{GRS}$ and $H_{GRS}$ are generator and parity check matrix of the same code, respectively.
Proof: If we solve the system in (1) for $v_i, v'_i$, then the solution is given by a non-zero element in the kernel of a Vandermonde matrix $V_{n,n-1}$ with $n = 1$ rows. Such a vector is given by $w_i$, as above, since $(w_1, \ldots, w_n)^T$ is the last column of the inverse Vandermonde matrix $V_{n,n}$ (cf. [24, eq. (9),(10)]). \hfill \blacksquare

Remark 1. Theorem 3 implies a method to compute the $v_i$ from the $v'_i$ or vice-versa, i.e.,

$$v_i = \frac{w_i}{v'_i} \quad \text{and} \quad v'_i = \frac{w_i}{v_i}$$

III. COEFFICIENT GROWTH OVER THE RATIONAL NUMBERS

In the following, we study the coefficient growth during computations over the rational numbers. The results can in principle be extended to a wider class of number fields by generalizing the following well-known notion of bitsize.

Definition 2 (Generalization of [25, p. 142]). Let $a$ be an element of one of the sets in $\{\mathbb{Z}, \mathbb{Q}, \mathbb{Q}[x], \mathbb{Q}^{k \times n}\}$. We define its 

- $a \in \mathbb{Z}$:
  $$\lambda(a) := \begin{cases} 
  \lfloor \log_2(|a|) \rfloor + 1, & \text{if } a \neq 0 \\
  0, & \text{if } a = 0
  \end{cases}$$

- $a = \frac{b}{c} \in \mathbb{Q}$ with $b, c \in \mathbb{Z}$, $c \neq 0$, and $\gcd(b, c) = 1$:
  $$\lambda(a) := \max\{\lambda(b), \lambda(c)\}.$$

- $a(x) = \sum_{i=0}^{n} a_i \cdot x^i \in \mathbb{Q}[x]$ with $a_i \in \mathbb{Z}$ and $b \in \mathbb{N} \setminus \{0\}$ such that $\gcd(a_0, \ldots, a_n, b) = 1$:
  $$\lambda(a(x)) := \max\{\lambda(a_0), \ldots, \lambda(a_n), \lambda(b)\}.$$

- $A = (a_{ij}) \in \mathbb{Q}^{k \times n}$:
  $$\lambda(A) = \max\{\lambda(a_{ij}) : i = 1, \ldots, k \text{ and } j = 1, \ldots, n\}.$$

Remark 2. Note that the bitsize does not necessarily cover the required memory space, e.g. for a rational number only the maximum of the bitsizes of numerator and denominator is regarded, which means, that for equal bitsizes of numerator and denominator the actual memory space is twice the bitsize.

Theorem 4 (Coefficient Growth [25, p. 142]). Let $a, b$ be polynomials in $\mathbb{Z}[x]$ with coefficients $a_i, b_j$ for $i \in \{0, \ldots, n := \deg(a(x))\} \text{ and } j \in \{0, \ldots, m := \deg(b(x))\}$ and $c, d \in \mathbb{Q}$, then the following statements hold:

1) $\lambda(a(x) + b(x)) \leq \max\{\lambda(a(x)), \lambda(b(x))\} + 1$

2) $\lambda(a(x) \cdot b(x)) \leq \lambda(a(x)) + \lambda(b(x)) + \lambda(\min\{n, m\}) + 1$

3) $\lambda(cd) \leq \lambda(c) + \lambda(d)$

4) $\lambda(\frac{c}{d}) \leq \lambda(c) + \lambda(d)$

5) $\lambda(c + d) \leq \lambda(c) + \lambda(d) + 1$

Proof: 1) For polynomials $a, b$ of degree 0, it’s trivial. For polynomials $a, b$ of arbitrary degree (w.l.o.g. we assume $n \geq m$ with coefficients $a_i, b_j$ for $i \in \{0, \ldots, n\}$ and $j \in \{0, \ldots, m\}$ it holds that

$$\lambda(a(x) + b(x)) = \max\{\lambda(a_0 + b_0), \ldots, \lambda(a_m + b_m), \lambda(a_{m+1}), \ldots, \lambda(a_n)\} \leq \max\{\max\{\lambda(a_0), \lambda(b_0)\} + 1, \ldots, \max\{\lambda(a_m), \lambda(b_m)\} + 1, \lambda(a_{m+1}), \ldots, \lambda(a_n)\} \leq \max\{\lambda(a_0), \ldots, \lambda(a_n), \lambda(b_0), \ldots, \lambda(b_m)\} + 1$$

$$= \max\{\max\{\lambda(a_0), \ldots, \lambda(a_n)\}, \max\{\lambda(b_0), \ldots, \lambda(b_m)\}\} + 1 = \max\{\lambda(a(x)), \lambda(b(x))\} + 1$$

2) $\lambda(a(x) \cdot b(x)) = \max_{k \in \{0, \ldots, n+m\}} \{\lambda(\sum_{i+j=k} a_i b_j)\}$

$$\leq \max\{\lambda(a_0, \ldots, a_n) \cdot \lambda(b_0, \ldots, b_m) \cdot (\min\{n, m\} + 1)\} + 1$$

$$= \log_2\{\max\{a_0, \ldots, a_n\} \cdot \max\{b_0, \ldots, b_m\} \cdot (\min\{n, m\} + 1)\} + 1$$

$$= \max\{\lambda(a_0, \ldots, a_n) + \lambda(\max\{b_0, \ldots, b_m\})\} + 1$$

$$= \lambda(\min\{n, m\} + 1) + 1$$

$$= \lambda(a(x)) + \lambda(b(x)) + \lambda(\min\{n, m\} + 1)$$

3) We write $c = \frac{c_1}{c_2}$ and $d = \frac{d_1}{d_2}$ with $c_1, c_2, d_1, d_2 \in \mathbb{Z}$ with \gcd\(c_1, c_2, d_1, d_2\) = \gcd\(d_1, d_2\) = 1. It holds:

$$\lambda(cd) = \lambda\left(\frac{c_1 \cdot d_1}{c_2 \cdot d_2}\right) \leq \max\{\lambda(c_1 \cdot d_1), \lambda(c_2 \cdot d_2)\}$$

$$= \max\{\log_2(|c_1|) + 1, \log_2(|c_2|) + 1\}$$

$$\leq \max\{\log_2(|c_1|) + 1 + \log_2(|d_1|) + 1, \log_2(|c_2|) + 1 + \log_2(|d_2|) + 1\}$$

$$\leq \max\{\lambda(c_1) + \lambda(d_1), \lambda(c_2) + \lambda(d_2)\} \leq \lambda(c) + \lambda(d).$$

4) Let $c' = c^{-1}$, then the claim follows by 3).

5) $\lambda(c + d) \leq \lambda(\frac{c_1 d_1 + c_2 d_2}{c_2 d_2}) = \max\{\lambda(\frac{c_1 d_2 + d_1 c_2}{c_2 d_2})\}$

$$\leq \max\{\lambda(\frac{c_1 d_1 \cdot d_2 + d_1 c_2}{c_2 d_2})\} \leq \max\{\lambda(c_1 d_1) + 1, \lambda(c_2 d_2)\}$$

$$= \lambda(c_1 d_1) + \lambda(c_2 d_2) + 1 = \lambda(c) + \lambda(d) + 1$$

Theorem 4 implies the following statements about coefficient growth in a vector or matrix multiplication.

Theorem 5 (Multiplication of vector and vector). Let $a, b \in \mathbb{Q}^n$. Then,

$$\lambda(ab^T) \leq n \cdot (\lambda(a) + \lambda(b) + 1).$$

Proof: Using Theorem 4, we obtain

$$\lambda(ab^T) = \lambda(\sum_{i=1}^{n} a_i b_i) \leq \sum_{i=1}^{n} (\lambda(a_i b_i)) + n - 1 \leq \sum_{i=1}^{n} (\lambda(a_i) + \lambda(b_i)) + n - 1,$$

which implies the claim. \hfill \blacksquare

\footnote{Set $k = \mu = n$, note that $\sigma_{1,n-1} = 1$.}
Theorem 6 (Multiplication of vector and matrix). Let $A \in \mathbb{Q}^{n \times r}$ and $B \in \mathbb{Q}^{r \times m}$. Then,

$$\lambda(AB) \leq r(\lambda(A) + \lambda(B) + 1).$$

**Proof**: The statement directly follows from Theorem 5 since any entry of the product $AB$ is the result of the multiplication of a row of $A$ with a column of $B$, which are both vectors of length $r$.

IV. COEFFICIENT GROWTH IN ENCODING RS CODES OVER THE RATIONAL NUMBERS

In this section we study the coefficient growth during encoding with a generator matrix $G$, i.e., we derive a bound on the bitsize of the codeword $c \in \mathbb{Q}^n$ obtained from an information word $u \in \mathbb{Q}^k$ with a given bitsize. We also show how to reduce the coefficient growth compared to the standard generator matrix given in Theorem 2.

**Theorem 7.** Let $c$ be an RS codeword generated by encoding $u \in \mathbb{Q}^k$ with generator matrix $G \in \mathbb{Q}^{k \times n}$. Then

$$\lambda(c) \leq k(\lambda(u) + \lambda(G) + 1).$$

**Proof**: The claim follows directly from Theorem 6.

Hence, the maximal coefficient growth depends heavily on the choice of the generator matrix. We can bound $\lambda(G)$ as follows.

**Corollary 1.** Let $\alpha_i, v'_i \in \mathbb{Q}\setminus\{0\}, \forall i = 1, \ldots, n$ and $G = (\alpha_1'v'_1, \ldots, \alpha_n'v'_n)$, $i = 1, \ldots, n, j = 0, \ldots, k - 1$ be a Vandermonde matrix multiplied with diag($v'_i$), where diag is a diagonal matrix with entries $v'_i$ as diagonal elements. Further let $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $v' = (v'_1, \ldots, v'_n)$. Then

$$\lambda(G) \leq (k - 1)\lambda(\alpha) + \lambda(v').$$

**Proof**: We have

$$\lambda(G) = \max_{i=1, \ldots, n} \{\lambda(\alpha_i'v'_i) : \alpha_i, v'_i \in \mathbb{Q}\}$$

$$\leq \max_{i=1, \ldots, n} \{\lambda(\alpha_i) + \lambda(v'_i) : \alpha_i, v'_i \in \mathbb{Q}\},$$

which implies the claim.

**Remark 3.** For $v'_i = 1, i = 1, \ldots, n$ the addition of $\lambda(v') = 1$ can be omitted, since then $\alpha_i'v'_i = \alpha_i' \forall i$. In this case, (2) is fulfilled with equality.

**Example 1.** If we choose $\alpha = (1, \ldots, n)$, $v' = (1, \ldots, 1)$ then $\lambda(G) = (k - 1)\lambda(n)$, whereas the allocated memory is

$$\sum_{j=1}^{k-1} \frac{n}{j} \log_2(i) + 1 \geq \sum_{j=1}^{k-1} \frac{n}{j} \log_2(i)$$

$$= \frac{(k - 1)k}{2} \log_2(n) = (k - 1)\frac{k}{2} \lambda(n)$$

$$\geq (k - 1)\lambda(n) \quad \text{for } k > 2.$$ 

This example shows, that the bitsize does not cover the necessary memory space.

According to [26, Theorem 1] the generator matrix of a GRS code can be brought to systematic form, where the identity matrix is followed by a Cauchy matrix, which is defined as follows.

**Theorem 8 (Cauchy Generator Matrices [26, Theorem 1]).**

Let $G_{\text{GRS}}$ as defined in Definition 1. The code has a systematic generator matrix of the form $G = (I_{k \times k} | A)$, where $I_{k \times k}$ is an identity matrix of size $k \times k$ and $A = \left(\frac{c_i d_j}{\alpha_i - \alpha_j}\right)$ is a Cauchy matrix with

$$a_i = \alpha_i, \quad i = 1, \ldots, k$$

$$b_j = \alpha_j + k, \quad j = 1, \ldots, n - k$$

$$c_i = (v'_i)^{-1} \prod_{1 \leq t \leq k, t \neq i} (\alpha_i - \alpha_t), \quad i = 1, \ldots, k$$

$$d_j = v'_j + \prod_{1 \leq t \leq k} (\alpha_j + \alpha_t), \quad j = 1, \ldots, n - k.$$ 

Using a generator matrix in systematic form, we obtain a generator matrix with a lower $\lambda(G)$ than a generator matrix in Vandermonde form.

**Corollary 2.** Let $G = (I_{k \times k} | A)$ be a generator matrix of $C_{\text{GRS}}$ as defined in Theorem 8, where the $v'_i$ are chosen such that $c_i d_j = 1$ are zero, we get

$$\lambda(G) \leq 2\lambda(\alpha) + 1.$$ 

**Proof**: The claim directly follows from the fact that in this case $A = \left(\frac{1}{\alpha_i - \alpha_j}\right)$ and Theorem 4.

We obtain the $v'_i$ for Corollary 2 as follows.

**Remark 4.** If $v_i \in H_{\text{GRS}}$ have the following form

$$v_i = \prod_{k+1 \leq t \leq n} (\alpha_i - \alpha_t)^{-1}$$

and $v'_i = \alpha_i$, from Theorem 3. Then the Cauchy matrix inside the corresponding generator matrix $G_{\text{GRS}}$ has the form

$$A_{ij} = \frac{1}{\alpha_i - \alpha_j + k},$$

i.e. $c_i$, $d_j$ from equations (5) and (6) fulfill $c_i d_j = 1$.

For arbitrary $v'_i$, we get a worse bound on $\lambda(G)$ if the Vandermonde matrix is in systematic form.

**Corollary 3.** If $G = (I_{k \times k} | A)$ be a generator matrix of $C_{\text{GRS}}$ as defined in Theorem 8, where $A = \left(\frac{c_i d_j}{\alpha_i - \alpha_j}\right) \in \mathbb{Q}^{k \times (n-k)}$, $i = 1, \ldots, k, j = 1, \ldots, n - k$, then

$$\lambda(G) \leq 2(2k - 1)\lambda(\alpha) + 2\lambda(v') + 2k - 1.$$ 

**Proof**: The claim directly follows when inserting equations (3) - (6) into the definition of the bitsize for matrices (see Def. 2) and the fact that $\lambda(1) = 1$, whereas the argument
of λ in the statement never gets zero. Then we have

\[
\lambda(G) \leq \max_{i=1, \ldots, k} \left\{ \lambda \left( \frac{v''_i + k}{v'_i} \prod_{t=1, t \neq i}^{n} (\alpha_{j + k} - \alpha_t) \right) \right\} \\
\leq \max_{i=1, \ldots, k} \left\{ \lambda(v'_i) + \lambda(v''_i) \right\} + 2 \lambda(v''_i) + \left( \frac{1}{2} - \sum_{i=1}^{n-1} \phi(i) + 1 \right) \\
\leq 2 \lambda(v''_i) + \left( \frac{1}{2} - \sum_{i=1}^{n-1} \phi(i) + 1 \right) \\
= 2 \lambda(v''_i) + \left( 4k - 2 \lambda(\alpha) + 2k - 1 \right).
\]

By the previous statements we can regard several cases of choices for \(v_i\) and \(v'_i\). The resulting upper bounds on the bitsizes of the generator and parity-check matrices are summarized in Table 1. More details about the formulas in the table can be found in the appendix.

All derived bounds depend on the bitsize of the code locators \(\alpha_i\). The following theorem shows how to choose such distinct \(\alpha_i\) with minimal \(\lambda(\alpha)\).

**Theorem 9.** If \(\alpha_1, \ldots, \alpha_n \in \mathbb{Q}\{0\}\) distinct, then \(\ell = \max\{\text{den}(\alpha_i), \text{num}(\alpha_i)\}\), where \(\text{num}(\alpha_i)\) is the numerator of \(\alpha_i\) and \(\ell \geq 4 \sum_{i=1}^{n} \phi(i) - 2\).

**Remark 5.** Using \(\alpha_i = i, i = 1, \ldots, n\) as evaluation points for an RS code as defined in Def. 1 leads to \(\lambda(c) \in O(\log n)\), whereas \(\alpha_i, \iota = 0, \ldots, n - 1\) as chosen by Theorem 9 for evaluation points will have \(\lambda(c) \in O(\log \sqrt{n})\). So asymptotically this only differs in a constant \(\frac{1}{2}\).

**Remark 6.** Since \(\lambda(\alpha) \in O(\log n)\), all the entries in Table 1 asymptotically belong to \(O(n \log n)\), except for \(c_d = 1\) for the Cauchy generator matrix, i.e. the case of Corollary 2, which is \(O(\log n)\). So this is asymptotically the best choice.

In Figure 1 the different choices of \(\alpha\) for Vandermonde and Cauchy generator matrices are compared. Here we chose \(v''_i = 1 \forall i = 1, \ldots, n\).

**V. COEFFICIENT GROWTH IN DECODING RS CODES OVER THE RATIONAL NUMBERS**

There are many decoding algorithms for GRS codes, both for decoding up to half-the-minimum distance [27], [28], [2], [29] and beyond [4], [5], [6], [7], [8].

In the following, we formulate the bounded-minimum-distance (BMD) decoder described in [3, Chapter 6], which is based on [27], [28], [2], [30], over \(\mathbb{Q}\) instead of a finite field (which is straightforward) and analyze its complexity (which is the involved part). For obtaining a good complexity, we use the variant based on the extended Euclidean algorithm (EEA), which was first suggested by [2]. For the core step of the algorithm, the EEA, we rely on an algorithm from [25], which is designed for small intermediate coefficient growth.

**A. Decoding Algorithm**

The decoding algorithm in [3, Chapter 6], which is described for finite fields there, returns the codeword \(c\) given only a received word \(r = c + e\). The algorithm finds the error positions \(E := \{i : e_i \neq 0\}\) by solving a key equation and then computes the error values \(e_i\) by Forney’s formula. All proofs showing the correctness of the algorithm do not make use of the finiteness of the field. Hence, the algorithm carries over directly and we only briefly recall its idea.

The key equation consists of the following polynomials.

- **Error locator polynomial**
  \[
  \Lambda(x) = \prod_{i \in E} (1 - \alpha_i x) \quad (\text{unknown at receiver}).
  \]
• Error evaluator polynomial
  \[ \Omega(x) = \sum_{i \in \mathcal{E}} e_i v_i \prod_{j \in \mathcal{E} \setminus \{i\}} (1 - \alpha_j x) \] (unknown at receiver).

• Syndrome polynomial
  \[ S(x) \equiv \sum_{i=0}^{d-2} s_i x^i := \sum_{i=0}^{d-2} \left( \sum_{j=1}^n r_j v_j \alpha_j^i \right) x^i \]
  \[ = \sum_{i=0}^{d-2} \left( \sum_{j=1}^n e_j v_j \alpha_j^i \right) x^i \] (known at receiver).

Note that \( [s_0, \ldots, s_{d-2}] = r \mathbf{H}_{GRS} \) is the syndrome with respect to the parity-check matrix \( \mathbf{H}_{GRS} \) of Definition 1.

The BMD decoder then finds the error locator and error evaluator polynomial by solving the following key equation.

**Lemma 1.** (Key Equation, generalization of [3, Equations (6.4)–(6.6)] to \( \mathbb{Q} \)) Let \( S(x), \Lambda(x), \Omega(x) \) be defined as above. Then,
\[
\Lambda(x)S(x) \equiv \Omega(x) \mod x^{d-1}, \\
\deg \Omega(x) < \deg \Lambda(x) = |\mathcal{E}|.
\]
*Proof:* The proof follows by exactly the same arguments as the statements in [3, Section 6.3], by replacing the finite field by \( \mathbb{Q} \).

The following lemma shows how to obtain the solution \( \Lambda(x), \Omega(x) \) of the key equation through the extended Euclidean algorithm \( \text{EEA}(a,b) \) over \( \mathbb{Q}[x] \), which computes for two input polynomials \( a, b \in \mathbb{Q}[x] \) with \( \deg a \geq \deg b \) a sequence \( (r_i, s_i, t_i) \) for \( i = 1, 0, 1, 2, \ldots \) such that we have
\[
r_i = s_i a + t_i b \quad \forall i
\]
and \( \deg r_i \) is strictly monotonically decreasing until \( r_i = 0 \).

For some integer \( t_{\text{stop}} \leq \deg b \), we denote by \( \text{EEA}(a,b,t_{\text{stop}}) \) the variant of the EEA that returns only the unique triple \( (r_h, s_h, t_h) \) for which we have
\[ \deg r_h < t_{\text{stop}} \leq \deg r_{h-1}. \]

**Lemma 2.** Let \( wt_2(e) \leq \left\lfloor \frac{d-1}{2} \right\rfloor \) and \( S(x) \) be the corresponding syndrome polynomial. Furthermore, let \( (r_h, s_h, t_h) \in \mathbb{Q}[x]^3 \) be the output of the EEA \( \text{EEA}(a,b,t_{\text{stop}}) \) with input
\[ a = \xi \cdot x^{d-1}, \quad b = \xi \cdot S(x), \quad \text{and} \quad t_{\text{stop}} = \frac{d-1}{2}, \]
where \( \xi \in \mathbb{Q} \) is a non-zero constant. Then, we have
\[ t_h = c \cdot \Lambda(x) \quad \text{and} \quad r_h = c \cdot \Omega(x), \]
where \( c \in \mathbb{Q} \) is a non-zero constant.

*Proof:* The proof follows from the same arguments that lead to [3, Corollary 6.5] since no step assumes the finiteness of the underlying field. The only difference besides the different field is the non-zero constant \( \xi \) here, which we will use for complexity reasons below, and which simply multiplies each \( (r_i, s_i, t_i), i \geq 0, \) in the EEA by \( \xi \).

The constant \( c \) in Lemma 2 can be determined as the \( 0^{\text{th}} \) coefficient of \( t_h \) since \( \Lambda_0 = 1 \) by definition.

**Lemma 3** (Forney’s formula). Let \( \Lambda'(x) = \sum_{i>0} i \Lambda_i x^{i-1} \) be the formal derivative of \( \Lambda(x) \). Then, we can compute the entries \( e_i \) of the error vector \( e \) by
\[
e_i = \frac{n}{\Lambda'(\alpha_i^{-1})} \left( \frac{d(\alpha_i^{-1})}{\Lambda(\alpha_i^{-1})} \right)
\]
for all \( i = 1, \ldots, n \).

*Proof:* The proof follows by the same arguments as in [3, Section 6.5].

Note that we get \( e_i = 0 \) for \( i \notin \mathcal{E} \), so there is no need to find \( \mathcal{E} \) separately. However, this can be done to reduce the runtime of the algorithm.

**B. Complexity Estimation**

For bounding the complexity of the algorithm, we need the following lemmas.

**Lemma 4.** Let \( \mathbf{H}_{GRS} \) as defined in Def. 1, \( s = (c+e)^\mathbf{H}_{GRS} \), \( \tau = \text{wt}_1(e) + \alpha, v \) as defined in Corollary 1.
\[ \lambda(s) \leq \tau (\lambda(e) + \lambda(\mathbf{H}_{GRS}) + 1). \]

*Proof:* Let \( \mathcal{E} = \{i_1, \ldots, i_r\} \) be error positions and \( e_\mathcal{E} = [e_{i_1}, \ldots, e_{i_r}] \) as well as \( \mathbf{H}_{\mathcal{E}} = [H_{i_1}, \ldots, H_{i_r}] \), where \( H_{i_r} \) is the \( i_r \)-th column of \( \mathbf{H}_{GRS} \). Then, we have
\[ \lambda(\mathbf{H}_{\mathcal{E}}) \leq \lambda(\mathbf{H}_{GRS}) \mathbf{H}_{GRS} = e \mathbf{H}_{\mathcal{E}} = e \mathbf{H}_{e}. \]

Thus, \( \lambda(s) = \lambda(e \mathbf{H}_{\mathcal{E}}) = \lambda(e, \mathbf{H}_{\mathcal{E}}) \leq \lambda(e, \mathbf{H}_{\mathcal{E}}) \leq \tau (\lambda(e) + \lambda(\mathbf{H}_{\mathcal{E}}) + 1). \]

Using the better coefficient growth bounds for computations over \( \mathbb{Z} \) (cf. Theorem 4), we obtain the following smaller bound in this case.

**Lemma 5.** Let \( \mathbf{H}_{GRS} \) from Def. 1 and \( \mathbf{H}_{GRS} \in \mathbb{Z}^{(n-k) \times n} \) (i.e., \( \alpha_i, v_i \in \mathbb{Z} \)), then
\[ \lambda(s) \leq (\tau + 1) \lambda(e) + \lambda(\mathbf{H}_{GRS}) + n. \]

*Proof:*

Using \( \lambda(\mathbf{H}) \leq \tau \lambda(e) \), we obtain
\[ \lambda(s) \leq \max\{\tau (\tau + 1) \lambda(e) + \lambda(\mathbf{H}) + (n - 1), \lambda(e) \} + 1 \]
\[ = (\tau + 1) \lambda(e) + \lambda(\mathbf{H}) + n. \]

**Lemma 6.** Let \( a, b \in \mathbb{Z}[x] \) with \( d_a = \deg a \geq d_b = \deg b \geq 1 \) and \( \lambda(a), \lambda(b) \leq t \). Then, \( \text{EEA}(a,b,t_{\text{stop}}) \) can be computed in
\[
O(d_a t^2 \log^2(d_a) (d_b^2 + \log^2(t)))
\]
bite operations using [25, Algorithm 6.57, page 188]. Furthermore, \( \text{EEA}(a,b,t_{\text{stop}}) \) can be computed in
\[
O(d_a t^2 \log^2(d_a) (d_b^2 + \log^2(t)))
\]
bite operations.

*Proof:* The statement follows directly from [25, Theorem 6.58], which is formulated in terms of the \( \infty \)-norm of the polynomials \( a \) and \( b \), i.e., \( ||a||_{\infty}, ||b||_{\infty} \leq A \) for some integer \( A \). This condition is equivalent to \( \lambda(a), \lambda(b) \leq \log_2(A) \). The complexity of the EEA with stopping condition follows from the fact that we save a factor \( d_b \) if only one specific triple \( (r_h, s_h, t_h) \) in the output sequence of the EEA is explicitly computed, cf. [25, page 188].

In the following, denote by \( \text{den}(\alpha) \) and \( \text{num}(\alpha) \) the (reduced) denominator and numerator of \( \alpha \in \mathbb{Q} \), respectively.

**Lemma 7.** Let \( S(x) \) be the syndrome polynomial and \( \xi = \text{lcm}(\text{den}(s_0), \ldots, \text{den}(s_{d-2})) \).
\[ \xi S(x) \in \mathbb{Z}[x] \quad \text{and} \quad \lambda(\xi S(x)) \leq d(\lambda(s) + 1). \]
Proof: The first claim, $\xi S(x) \in \mathbb{Z}[x]$ is obviously fulfilled (in fact, $\xi$ is the smallest positive integer with this property). For the bitsize, we have

$$\lambda(\xi) \leq \lambda\left(\prod_{i=0}^{d-2} \text{den}(s_i)\right) \leq \sum_{i=0}^{d-2} \lambda(s_i) + d - 1$$

$$\leq (d - 1)(\lambda(s) + 1),$$

Lemma 8. For $|E| = \tau$ and $\Lambda(x) = \prod_{i \in E} (1 - \alpha_i x)$ and $\sum_{j=0}^{\tau} \Lambda_j x^j$ it holds that

$$\lambda(\Lambda(x)) \leq \tau(\lambda(\alpha) + 2).$$

Proof: Let $\zeta := \prod_{i \in E} \text{den}(a_i)$. Then,

$$\Lambda(x) = \frac{1}{\zeta} \prod_{i \in E} \left[\text{den}(a_i) - \text{num}(\alpha_i) x\right],$$

where $a_i(x) \in \mathbb{Z}[x]$, $\deg a_i(x) = 1$, and $\lambda(\alpha_i) \leq \lambda(\alpha_i)$.

By applying Property 2 of Theorem 4 inductively, we get

$$\lambda\left(\prod_{i \in E} a_i(x)\right) \leq \sum_{i \in E} \lambda(\alpha_i) + \tau \lambda(2) \leq \tau(\lambda(\alpha) + 2).$$

Due to $\zeta \in \mathbb{N}$ and $\lambda(\zeta) \leq \sum_{i \in E} \lambda(\alpha_i)$, the claim follows by $\lambda(\Lambda(x)) \leq \max\{\lambda(\prod_{i \in E} a_i(x)), \lambda(\zeta)\}$. $\blacksquare$

Lemma 9. For $\Omega(x) = \sum_{j \in E} e_j v_j \prod_{i \in E \setminus \{j\}} (1 - \alpha_i x)$ and $|E| = \tau$ it holds that

$$\lambda(\Omega(x)) \leq \tau(\lambda(\alpha) + \lambda(e) + \lambda(v) + 5).$$

Proof: Let $\zeta := \prod_{i \in E} \text{den}(a_i)$, $\eta := \prod_{i \in E} \text{den}(e_i, v_i)$ and $a_i(x) := \text{den}(a_i) - \text{num}(\alpha_i) x$ then

$$\Omega(x) = \frac{1}{\zeta \eta} \sum_{i \in E} \text{num}(e_i, v_i) \text{den}(a_i) \prod_{j \in E \setminus \{i\}} \text{den}(e_j, v_j) a_j(x)$$

Since $\zeta \eta \cdot \Omega(x) \in \mathbb{Z}[x]$ it holds that

$$\lambda(\zeta \eta \cdot \Omega(x)) \leq \max_{i \in E} \lambda\left[\text{num}(e_i, v_i) \text{den}(a_i) \prod_{j \in E \setminus \{i\}} \text{den}(e_j, v_j) a_j(x)\right] + \tau - 1$$

$$\leq \max_{i \in E} \left[\lambda(\text{num}(e_i)) + 1 + \lambda(\text{num}(v_i)) + 1 + \lambda(\text{den}(a_i)) + 1 + \sum_{j \in E \setminus \{i\}} \lambda(\text{den}(e_j)) + \tau - 1 + \sum_{j \in E \setminus \{i\}} \lambda(\text{den}(v_j) + \tau - 1\right]$$

$$+ 1 + \sum_{j \in E \setminus \{i\}} \lambda(\alpha_j(x)) (\tau - 1)\lambda(\alpha) + 2$$

$$+ \tau - 1$$

$$\leq \lambda(\alpha) + \lambda(e) + \lambda(v) + (\tau - 1)\lambda(e) + (\tau - 1)\lambda(v) + 5\tau$$

$$= \tau(\lambda(\alpha) + \lambda(e) + \lambda(v) + 5).$$

Also for the bitsize of $\zeta \eta$ we have that

$$\lambda(\zeta \eta) = \lambda(\prod_{i \in E} \text{den}(e_i, v_i, \alpha_i)) \leq \tau(\lambda(\alpha) + \lambda(e) + \lambda(v)).$$

And therefore the claim holds.

Lemma 10. Let $a(x) \in \mathbb{Q}[x]$ and $\alpha_i \in \mathbb{Q}$. Then, the evaluation of $a(x)$ can be implemented in

$$O^\ast(\deg(a(x)) \max\{\lambda(\alpha_i), \lambda(a(x))\}).$$

bit operations.

Proof: Applying Horner’s rule [25, p. 101] for evaluation, one gets $\deg a(x)$ multiplications and $\deg a(x)$ additions. By naive computation of the additions and the multiplications as in [31], the complexity for integers of bitsize $t_1$ and $t_2$ (define: $n_0 = \max\{t_1, t_2\}$) is $O(n_0)$, respectively $O(n_0 \log n_0 \log \log n_0)$ (cf. Algorithms 2.1 in [25]). Hence,

$$O(\deg(a(x)) (\log \log \log n + 1)) = O^\ast(\deg(a(x)) \max\{\lambda(\alpha_i), \lambda(a(x))\}).$$

Corollary 4. $\Lambda'(\alpha_i^{-1})$ and $\Omega(\alpha_i^{-1})$, and thus the $i$th error value $e_i$ can be computed in

$$O^\ast(\tau^2(\lambda(\alpha) + \lambda(e) + \lambda(v)))$$

bit operations.

C. Summary of the Algorithm

Algorithm 1: Decoding Algorithm for GRS Codes over $\mathbb{Q}$

Input: Received word $r = c + e$, where $c \in \mathbb{C}_{\text{GRS}}$ and $\text{wt}_{\text{H}}(e) \leq \frac{n - d}{2}$.

Output: Codeword $c'$

1. $s \leftarrow rH_{\text{GRS}}$
2. $S(x) \leftarrow \sum_{i=0}^{d-2} s_i x^i$
3. $\xi \leftarrow \text{lcm}(\text{den}(s_0), \ldots, \text{den}(s_{d-2}))$
4. $(r_0, s_0, t_0) \leftarrow \text{EEA}(\xi \cdot S(x), \xi \cdot x^{d-1}, \frac{d-1}{2})$
5. $c \leftarrow 0^\text{th}$ coefficient of $t_0$
6. $(\Lambda(x), \Omega(x)) \leftarrow e^{-1}(r_0, t_0)$
7. $\lambda'(x) \leftarrow \sum_{i=0}^{\tau} \lambda'(x_i^{-1})$
8. $e_i \leftarrow \frac{\text{den}(e_i)}{\text{num}(e_i)}$ for $i = 1, \ldots, n$
9. return $c = r - e$

Theorem 10. Algorithm 1 returns the correct codeword if $\text{wt}_{\text{H}}(e) \leq \frac{n - d}{2}$. Its complexity in bit operations is

$$O^\ast\left(d^2 \left[\lambda(e) + \lambda(H_{\text{GRS}})\right]^2\right).$$

Proof: Correctness follows directly by Lemmas 2 and 3 above. The bottleneck of the algorithm is the EEA, which can be implemented by [25, Algorithm 6.57, page 188]. The input polynomials, $a(x) = \xi \cdot x^{d-1}$ and $b(x) = \xi \cdot S(x)$ have bitsize $\lambda(a(x)), \lambda(b(x)) \leq d(\lambda(s) + 1) \leq d\tau(\lambda(e) + \lambda(H_{\text{GRS}}) + 2)$

$$O(d^2 \left[\lambda(e) + \lambda(H_{\text{GRS}})\right])$$

by Lemmas 7 and 4, and degrees $d_a = d - 1$ and $d_b \leq d - 2$. By Lemma 6, we can therefore bound the cost of Line 4 by

$$O\left(d^2 \left[\lambda(e) + \lambda(H_{\text{GRS}})\right]^2\cdot \log^2(d) \log \left(d^2 \left[\lambda(e) + \lambda(H_{\text{GRS}})\right]\right)\right).$$

The remaining steps of Algorithm 1 are negligible compared to this step due to Corollary 4.

For a special choice of $\alpha$, the following corollaries hold. We state it in terms of $n$ as we often have $d \in \Theta(n)$. $\blacksquare$

\footnote{O^\ast$ neglects logarithmic terms of its arguments}
Corollary 5. If the error $e$ has bitsize at most $t$ and the code locators are chosen with $\lambda(\alpha) \in O(\log(n))$ (e.g., $\alpha_i = i$ for $i = 1, \ldots, n$), then Algorithm 1 can be implemented in $O^\sim(\max\{n^7t^2, n^3\})$ bit operations. In case of $H_{\text{GRS}} \in \mathbb{Z}^{(n-k)\times n}$ we get $O^\sim(\max\{n^7t^2, n^3\}) = O^\sim(n^7t^2)$.

VI. TRADE-OFF BETWEEN ENCODING AND DECODING

Corollary 5 states that the complexity might be reduced by choosing the entries of $H_{\text{GRS}}$ in $\mathbb{Z}$. The next corollary also compares this reduction.

Corollary 6. Regard the setting of Lemma 4.

- For the case of $v_i = 1 \forall i = 1, \ldots, n$, we have
  \[ \lambda(s) \leq \tau(\lambda(e) + (n - k - 1)\lambda(\alpha) + 1). \]

- For the case $c_id_j = 1$, we have
  \[ \lambda(s) \leq \tau(\lambda(e) + 3(n - k)\lambda(\alpha) + n - k + 1). \]

- For the case $H_{\text{GRS}} \in \mathbb{Z}^{(n-k)\times n}$ and $v_i = 1 \forall i = 1, \ldots, n$, we have
  \[ \lambda(s) \leq (\tau + 1)\lambda(e) + (n - k - 1)\lambda(\alpha) + n. \]

- For the case $H_{\text{GRS}} \in \mathbb{Z}^{(n-k)\times n}$ and $c_id_j = 1$, we have
  \[ \lambda(s) \leq (\tau + 1)\lambda(e) + 3(n - k)\lambda(\alpha) + 2n - k. \]

Proof: Insert the results from Table I into $\lambda(H_{\text{GRS}})$.

In this case, however, we must have $v_i \in \mathbb{Z}$, which might yield large values of $\lambda(G)$ according to Theorem 3. This again can increase the coefficient growth during encoding. Hence, one can find a tradeoff between decoding complexity and coefficient growth during encoding by choosing $v_i$. Table II shows a part of the tradeoff by comparing the upper bounds for the bitsize of syndrome and codeword, that are derived in Theorem 7 and Lemma 4 in the sections before. Since $\lambda(\alpha) \in O(\log(n))$, we get $\lambda(c) \in O(n^2 \log(n))$ for all cases except for the Cauchy generator matrix $G_{\text{Cauchy}}$ in combination with $c_id_j = 1$, where we have $\lambda(c) \in O(n \log(n))$. Hence, this choice is asymptotically the best, since the syndrome has $\lambda(s) \in O(n^2 \log(n))$ anyway.

VII. CONCLUSION

In this paper we studied RS codes over arbitrary fields and analyzed the coefficient growth for encoding and decoding over the rational numbers. By deriving bounds on the intermediate coefficient growth during computations, we were able to show that decoding up to half-the-minimum distance is possible in a polynomial number of bit operations in the code length and bitsize of the error values.

As future work, the results can be extended to a wider class of number fields. Furthermore, the methods can be used to determine the bit complexity of the algorithms in [18], [19] for decoding Gabidulin codes over characteristic zero.

VIII. APPENDIX

Proofs for formulas in Table I. Case: $c_id_j = 1$, then $\lambda(G_{\text{GRS}}) \leq (k - 1)(3\lambda(\alpha) + 1)$

Proof:

\[ \lambda(v') \leq \max_{i = 1, \ldots, n} \left( \lambda(\prod_{1 \leq \lambda \leq \lambda_i} (\alpha_i - \alpha_{i+1})) \right) \]
\[ \leq \max_{i = 1, \ldots, n} \left( \sum_{\lambda = 1, \ldots, k} (\lambda(\alpha_i) + \lambda(\alpha_{i+1} + 1)) \right) \]
\[ \leq (k - 1) \max_{i = 1, \ldots, n} (2\lambda(\alpha_i) + 1) \]
\[ = (k - 1)(2\lambda(\alpha) + 1). \]

Case: $v_i = 1, v_i = w_i$, then $\lambda(G_{\text{GRS}}) \leq (2n + k - 3)\lambda(\alpha) + n - 1$

Proof: Inserting the condition into $A$ gives

\[ A_{ij} = (\alpha_i - \alpha_j - k)^{-1} \prod_{k+1 \leq \lambda \leq n} (\alpha_j + k - \alpha_{i+1})^{-1} \]
\[ \cdot \prod_{k+1 \leq \lambda \leq k} (\alpha_i - \alpha_{i+1})^{-1} \]
\[ \lambda(G_{\text{Cauchy}}) \leq \max_{i = 1, \ldots, n} \left( \sum_{\lambda = 1, \ldots, k} \lambda(\alpha_j + k - \alpha_{i+1}) \right) \]
\[ + \sum_{t = k+1, \ldots, n} \lambda(\alpha_i - \alpha_{i+1} + \lambda(\alpha_i) + \lambda_j + 1) \]
\[ \leq 2(n - k)(2\lambda(\alpha) + 1) + 2\lambda(\alpha) + 1. \]

Case: $v_i = 1, v_i = w_i$, then $\lambda(G_{\text{GRS}}) \leq (2n + k - 3)\lambda(\alpha) + n - 1$

Proof:

\[ \lambda(G_{\text{Cauchy}}) \leq \max_{i = 1, \ldots, n} \left( \lambda(\prod_{1 \leq \lambda \leq \lambda_i} (\alpha_j + k - \alpha_{i+1})) \right) \]
\[ \cdot \prod_{1 \leq \lambda \leq k} (\alpha_i - \alpha_{i+1})^{-1} \]
\[ \leq (k - 1)(2\lambda(\alpha) + 1) + (k - 1)(2\lambda(\alpha) + 1) \]

Case: $v_i = 1, v_i = w_i$, then $\lambda(H_{\text{GRS}}) \leq (3n - k - 3)\lambda(\alpha) + n - 1$

Proof:

\[ \lambda(v) \leq \max_{i = 1, \ldots, n} \left( \lambda(\prod_{t = 1, \ldots, n} (\alpha_i - \alpha_{t+1})) \right) \]
\[ \leq (n - 1)(2\lambda(\alpha) + 1). \]
TABLE II

| General            | $G = G_{GRS}$ | $G = G_{Cauchy}$ |
|--------------------|---------------|------------------|
| $c_i d_j = 1$      | $k(\lambda(u) + 3(k-1)\lambda(\alpha) + k)$ | $k(\lambda(u) + 2\lambda(\alpha) + 2)$ |
| $v_i = 1$, $v'_i = v_i$ | $k(\lambda(u) + (2n+k-3)\lambda(\alpha) + n)$ | $k(\lambda(u) + (2n-2k+1)(2\lambda(\alpha) + 1) + 1)$ |
| $\lambda(\epsilon) \leq k(\lambda(u) + \lambda(G) + 1) + 1$ | $\lambda(\sigma) \leq \tau(\lambda(\epsilon) + \lambda(H_{GRS}) + 1)$ |

Case $c_i d_j = 1$, then

$\lambda(G_{GRS}) \leq (3n - 3k - 1)\lambda(\alpha) + n - k$

Proof: Representation of $v_i$, see Remark 4.

$\lambda(v) \leq \max_{i=\nu} \{ \lambda(\prod_{1\leq t \leq n-k}(e_{k} - \alpha_{t+k})) \}$

\[ \leq (n - k)(2\lambda(\alpha) + 1) \]

REFERENCES

[1] I. S. Reed and G. Solomon, “Polynomial codes over certain finite fields,” SIAM J. Appl. Math., vol. 8, no. 2, pp. 300–304, 1960.
[2] Y. Sagiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, “A Method for Solving Key Equation for Decoding Goppa Codes,” Information and Control, vol. 27, no. 1, pp. 87–99, 1975.
[3] R. M. Roth, Introduction to Coding Theory. Cambridge UP, 2006.
[4] [7] M. Sudan, “Decoding of Reed–Solomon Codes Beyond the Error-Correction Bound,” J. Complexity, vol. 13, no. 1, pp. 180–193, 1997.
[5] V. Guruswami and M. Sudan, “Improved Decoding of Reed-Solomon and Algebraic-Geometric Codes,” in IEEE FOCS, 1998, pp. 26–37.
[6] Y. Wu, “New List Decoding Algorithms for Reed-Solomon and BCH Codes,” IEEE Trans. Inf. Theory, vol. 54, no. 8, pp. 3611–3630, 2008.
[7] 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, vol. 56, no. 10, pp. 5245–5252, 2010.
[8] J. Rosenkilde, “Power Decoding Reed–Solomon Codes up to the Johnson Radius,” AMC, vol. 12, no. 1, pp. 81–106, 2018.
[9] J. K. Wolf, “Redundancy, the Discrete Fourier Transform, and Impulse Noise Cancellation,” IEEE Trans. Commun., vol. 31, no. 3, pp. 458–461, Mar. 1983.
[10] T. J. Marshall, “Coding of Real-Number Sequences for Error Correction: A Digital Signal Processing Problem,” IEEE J. Select. Areas Commun., vol. 2, no. 2, pp. 381–392, Mar. 1984.
[11] W. Henkel, Zur Decodierung algebraischer Blockcodes über komplexen Alphabeten, ser. Fortschritt-Berichte VDI. VDI-Verlag, 1989.
[12] P. Parvaresh and B. Hassibi, “Explicit Measurements With Almost Optimal Thresholds for Compressed Sensing,” in IEEE Int. Conf. Acoust., Speech, Signal Process., Mar. 2008, pp. 3853–3856.
[13] M. Mohamed, S. Rizkalla, H. Zörlein, and M. Bossert, “Deterministic Compressed Sensing with Power Decoding for Complex Reed-Solomon Codes,” in ITG SCC, Feb 2015.
[14] H. Zörlein, “Channel Coding Inspired Contributions to Compressed Sensing,” Ph.D. dissertation, Ulm University, 2015. [Online]. Available: http://vts.uni-ulm.de/doc.asp?id=9748
[15] M. H. Mohamed, S. Puchinger, and M. Bossert, “Guruswami-Sudan List Decoding for Complex Reed–Solomon Codes,” in ITG SCC, 2017.
[16] R. M. Roth, “Tensor Codes for the Rank Metric,” IEEE Trans. Inf. Theory, vol. 42, no. 6, pp. 2146–2157, 1996.
[17] D. Augot, P. Loidreau, and G. Robert, “Rank metric and gabidulin codes in characteristic zero,” in Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on. IEEE, 2013, pp. 509–513.
[18] G. Robert, “A Quadratic Welch-Berlekamp Algorithm to Decode Generalized Gabidulin Codes, and Some Variants,” in IEEE ISIT, 2016.
[19] S. Müelich, S. Puchinger, D. Mödinger, and M. Bossert, “An Alternative Decoding Method for Gabidulin Codes in Characteristic Zero,” in IEEE ISIT, 2016.
[20] D. Augot, P. Loidreau, and G. Robert, “Generalized Gabidulin Codes Over Fields of Any Characteristic,” Des. Codes Cryptogr., pp. 1–42, 2017.
[21] G. Robert, “A New Constellation for Space-Time Coding,” in WCC, 2015.
[22] S. Müelich, S. Puchinger, and M. Bossert, “Low-rank matrix recovery using gabidulin codes in characteristic zero,” Electronic Notes in Discrete Mathematics, vol. 57, pp. 161–166, 2017.
[23] C. Sippel, C. Ott, S. Puchinger, and M. Bossert, “Reed–Solomon Codes over Fields of Characteristic Zero,” Extended version of this paper, https://nt.uni-ulm.de/sippelottpuchrs2019extended, 2019.
[24] N. Macon and A. Spitzbart, “Inverses of Vandermonde matrices,” The American Mathematical Monthly, vol. 65, no. 2, pp. 95–100, 1958.
[25] J. von zur Gathen and J. Gerhard, Modern Computer Algebra. Cambridge university press, 2013.
[26] R. M. Roth and G. Seroussi, “On Generator Matrices of MDS Codes,” IEEE Trans. Inf. Theory, vol. 51, no. 6, pp. 826–830, Nov. 1985.
[27] D. Gorenstein and N. Zierler, “A Class of Error-Correcting Codes in p’m Symbols,” SIAM J. Appl. Math, vol. 9, no. 2, pp. 207–214, 1961.
[28] W. Peterson, “Encoding and Error-Correction Procedures for the Bose-Chaudhuri Codes,” IRE Trans. Inf. Theory, vol. 6, no. 4, pp. 459–470, 1960.
[29] L. R. Welch and E. R. Berlekamp, “Error Correction for Algebraic Block Codes,” Dec. 30 1986, us Patent 4,633,470.
[30] G. Forney, “On decoding BCH codes,” IEEE Trans. Inf. Theory, vol. 11, no. 4, pp. 549–557, 1965.
[31] A. Schönhage and V. Strassen, “Schnelle Multiplikation Großer Zahlen,” Computing, vol. 7, no. 3-4, pp. 281–292, 1971.