Complexity of Decoding Gabidulin Codes

Maximilien Gadouleau and Zhiyuan Yan
Department of Electrical and Computer Engineering
Lehigh University, PA 18015, USA
E-mails: {magc, yan}@lehigh.edu

***Abstract*— In this paper, we analyze the complexity of decoding Gabidulin codes using the analogs in rank metric codes of the extended Euclidean algorithm or the Berlekamp-Massey algorithm. We show that a subclass of Gabidulin codes reduces the complexity and the memory requirements of the decoding algorithm. We also simplify an existing algorithm for finding roots of linearized polynomials for decoding Gabidulin codes. Finally we analyze and compare the asymptotic complexities of different decoding algorithms for Gabidulin codes.**

I. Introduction

Error correction codes with the rank metric [1]–[5] have been receiving steady attention in the literature due to their applications in storage systems [3], public-key cryptosystems [4], space-time coding [5], and network coding [6], [7].

The pioneering works in [1]–[3] have established many important properties of rank metric codes. Independently in [1]–[3], a Singleton bound (up to some variations) on the minimum rank distance of codes was established, and a class of codes that achieve the bound with equality was constructed. We refer to codes that attain the Singleton bound as maximum rank distance (MRD) codes, and the class of MRD codes proposed in [2] as Gabidulin codes henceforth. In [1], [2], analytical expressions to compute the weight distribution of linear MRD codes were also derived. In [3], [8], it was shown that Gabidulin codes are also optimal in the sense of a Singleton bound in crisscross weight, a metric considered in [3], [9] for crisscross errors, which occur in storage devices. In [6], [7], a class of asymptotically optimal codes for error and erasure correction in random network coding was designed based on Gabidulin codes. Following the works in [1]–[3], the construction in [2] was extended in [10] and the properties of subspace subcodes and subfield subcodes were considered in [11]; the error performance of Gabidulin codes was investigated in [9], [12].

Gabidulin codes can be viewed as evaluations of linearized polynomials, a special class of polynomials over finite fields [13], [14]. These polynomials form an algebra under addition and symbolic product, and hence have an extended Euclidean algorithm (EEA). In [2], a bounded rank distance decoder for Gabidulin codes was designed based on the EEA for linearized polynomials. Berlekamp [15] designed a decoding algorithm for Reed-Solomon codes, which can be interpreted as the design of a minimum length linear feedback shift register [16]. This algorithm was adapted to the decoding of Gabidulin codes in [17], [18], where it was extended to correct both errors and erasures. A decoding algorithm that parallels the Peterson-Gornstein-Zierler (PGZ) algorithm was introduced for Gabidulin codes in [3]. Finally, the counterpart of the Welch-Berlekamp (WB) algorithm was considered in [19]. Henceforth, we omit "the counterpart" in our references for simplicity.

The complexity of decoding Gabidulin codes using the PGZ algorithm or the WB algorithm was analyzed in [3] and [19], respectively. In this paper, we analyze the complexity of the decoding algorithms for Gabidulin codes using the EEA or the Berlekamp-Massey algorithm (BMA). We first investigate the complexity of operations for linearized polynomials. We also consider the subclass of Gabidulin codes for which the parity check matrix is generated by elements of a normal basis. We show that this subclass of Gabidulin codes reduces the complexity and the memory requirements of the algorithm. The most efficient algorithm so far for finding roots of a linearized polynomial, essential to the decoding of Gabidulin codes, was given in [20]. We simplify this algorithm for decoding Gabidulin codes. We finally compare the complexities of these decoding algorithms with other existing algorithms.

The rest of the paper is organized as follows. Section II gives a brief review of the rank metric, linearized polynomials, and Gabidulin codes in order to keep this paper self-contained. In Section III, we investigate the complexity of linearized polynomial operations. In Section IV, we determine the complexity of the EEA and the BMA for linearized polynomials. Section V investigates the complexity of the rest of the decoding algorithm. Finally, Section VI compares the asymptotic complexities of different decoding algorithms for Gabidulin codes.

II. Preliminaries

A. Rank metric

Consider an n-dimensional vector $\mathbf{x} = (x_0, x_1, \ldots, x_{n-1}) \in \mathrm{GF}(q^m)^n$. The field $\mathrm{GF}(q^m)$ may be viewed as an m-dimensional vector space over $\mathrm{GF}(q)$. The rank weight of $\mathbf{x}$, denoted as $\mathrm{rk}(\mathbf{x})$, is defined to be the *maximum* number of coordinates in $\mathbf{x}$ that are linearly independent over $\mathrm{GF}(q)$ [2]. The coordinates of $\mathbf{x}$ thus span a linear subspace of $\mathrm{GF}(q^m)$ with dimension equal to $\mathrm{rk}(\mathbf{x})$.

For all $\mathbf{x}, \mathbf{y} \in \mathrm{GF}(q^m)^n$, it is easily verified that $d_{\mathrm{R}}(\mathbf{x}, \mathbf{y}) \stackrel{\mathrm{def}}{=} \mathrm{rk}(\mathbf{x} - \mathbf{y})$ is a metric over $\mathrm{GF}(q^m)^n$, referred to as the *rank metric* henceforth [2]. The *minimum rank distance* of a code C, denoted as d_{R}, is simply the minimum rank distance over all possible pairs of distinct codewords. The minimum rank

978-1-4244-2247-0/08/$25.00 ©2008 IEEE.

distance d_{R} of a code of length n over $\mathrm{GF}(q^m)$ satisfies $d_{\text{R}} \leq d_{\text{H}}$ [2], where d_{H} is the minimum Hamming distance of the same code. Due to the Singleton bound on the minimum Hamming distance of block codes [21], the minimum rank distance of a block code of length n and cardinality M over $\mathrm{GF}(q^m)$ thus satisfies

$$d_{\text{R}} \leq n - \log_{q^m} M + 1. \tag{1}$$

In this paper, we refer to the bound in (1) as the Singleton bound for rank metric codes and codes that attain the equality as MRD codes. The rank distribution of linear MRD codes was determined in [1], [2].

B. Linearized polynomials

Linearized polynomials (LP's) were first introduced in [13], and have since been widely studied (see [21]–[23]). In order to simplify notations, we denote $[i] \stackrel{\text{def}}{=} q^i$ henceforth.

Definition 1 (Linearized polynomial): A linearized polynomial $F(z)$ over $\mathrm{GF}(q^m)$ is a polynomial of the form $F(z) = \sum_{i=0}^{u} f_i z^{[i]}$, where $f_i \in \mathrm{GF}(q^m)$ for $0 \leq i \leq u$.

We refer to u as the *degree* of $F(z)$. Note that for any $\alpha \in \mathrm{GF}(q^m)$, $\alpha^{[m]} = \alpha^{[0]}$, and hence we can always assume that the degree of $F(z)$ satisfies $u < m$. Linearized polynomials can be viewed as linear operators over $\mathrm{GF}(q^m)$, thus their roots form a linear subspace of $\mathrm{GF}(q^m)$ with dimension at most equal to the degree of the LP. Linearized polynomials form an algebra under the addition and the symbolic product, defined as $L(z) * M(z) \stackrel{\text{def}}{=} L(M(z))$. Note that the symbolic product is not commutative. There thus exist both the left- and right-long divisions for LP's. There exists an extended Euclidean algorithm (EEA) in this algebra, which is similar to the EEA for polynomials, for either left- or right-division.

C. Gabidulin codes and their decoding

A class of linear MRD codes, referred to as Gabidulin codes henceforth, were defined independently in [1]–[3]. Let $n \leq m$ and $h_0, h_1, \ldots, h_{n-1} \in \mathrm{GF}(q^m)$ be linearly independent. A Gabidulin code is a linear code of length n, dimension $n-d+1$, and minimum rank distance d with the following parity-check matrix

$$\mathbf{H} = \begin{pmatrix} h_0 & h_1 & \ldots & h_{n-1} \\ h_0^{[1]} & h_1^{[1]} & \ldots & h_{n-1}^{[1]} \\ \vdots & \vdots & \ddots & \vdots \\ h_0^{[d-2]} & h_1^{[d-2]} & \ldots & h_{n-1}^{[d-2]} \end{pmatrix}. \tag{2}$$

We now review the problem of decoding Gabidulin codes. Let $\mathcal{C}$ be an $(n, k, d = n-k+1)$ Gabidulin code over $\mathrm{GF}(q^m)$ with parity-check matrix $\mathbf{H} = (h_j^{[i]})_{i,j=0}^{d-2,n-1}$. Suppose we receive $\mathbf{y} = \mathbf{c} + \mathbf{e} \in \mathrm{GF}(q^m)^n$, where $\mathbf{c} \in \mathcal{C}$ and $\mathrm{rk}(\mathbf{e}) = r \leq \lfloor \frac{d-1}{2} \rfloor$. The objective is to determine $\mathbf{e} = (e_0, e_1, \ldots, e_{n-1})$. We denote $\mathbf{e} = (E_0, E_1, \ldots, E_{r-1})\mathbf{Y}$, where $E_0, E_1, \ldots, E_{r-1} \in \mathrm{GF}(q^m)$ are linearly independent and $\mathbf{Y} \in \mathrm{GF}(q)^{r \times n}$ has full rank. We also define $\mathbf{X} \stackrel{\text{def}}{=} \mathbf{Y}\mathbf{H}^T = (x_i^{[j]})_{i,j=0}^{r-1,d-2}$.

The decoding algorithm of Gabidulin codes based on the EEA or the BMA can be split into six steps [2].

- **Step 1** Calculate the syndrome $\mathbf{s} \stackrel{\text{def}}{=} \mathbf{y}\mathbf{H}^T = (s_0, s_1, \ldots, s_{d-2}) \in \mathrm{GF}(q^m)^{d-1}$ and the associated linearized polynomial $S(z) = \sum_{i=0}^{d-2} s_i z^{[i]}$.
- **Step 2** Determine $\Lambda(z)$ and $F(z)$ such that $\deg F(z) < r$ and $F(z) = \Lambda(z) * S(z) \mod z^{[d-1]}$ using either the EEA or the BMA.
- **Step 3** Determine r linearly independent roots $E_0, E_1, \ldots, E_{r-1}$ of $\Lambda(z)$.
- **Step 4** Determine $\mathbf{x} = (x_0, x_1, \ldots, x_{r-1})$ using $\sum_{j=0}^{r-1} E_j x_j^{[p]} = s_p$ for $0 \leq p \leq r-1$.
- **Step 5** Determine $\mathbf{Y}$ using $x_p = \sum_{j=0}^{n-1} Y_{p,j} h_j$ for $0 \leq p \leq r-1$.
- **Step 6** Calculate $\mathbf{e} = (E_0, E_1, \ldots, E_{r-1})\mathbf{Y}$.

III. Complexity of linearized polynomial operations

All finite field operations are over $\mathrm{GF}(q^m)$ unless specified otherwise. An operation over $\mathrm{GF}(q^m)$ can be easily be implemented in $O(m^2)$ operations over $\mathrm{GF}(q)$. However, using more sophisticated techniques, these operations can be implemented in fewer operations over $\mathrm{GF}(q)$.

We first discuss the representation of the finite field elements. The elements of $\mathrm{GF}(q^m)$ are usually stored as m-dimensional vectors over $\mathrm{GF}(q)$ with regard to some basis. A normal basis is most suitable, since all the power elevations of the type $\alpha^{[i]}$ are performed using cyclic shifts, which we will ignore in our complexity analysis. We thus assume that a normal basis $B = \{\beta, \beta^{[1]}, \ldots, \beta^{[m-1]}\}$ is used to represent the elements of $\mathrm{GF}(q^m)$ henceforth.

We consider two LP's $F(z) = \sum_{i=0}^{u} f_i z^{[i]}$ of degree u and $G(z) = \sum_{j=0}^{v} g_j z^{[j]}$ of degree v, where $0 \leq v \leq u < m$.

Clearly, the sum $F(z)+G(z)$ requires at least $\min\{u+1, v+1\}$ additions. Also, if the values of u and v are unknown, the sum can be done with m additions.

The symbolic product $H(z) = F(z) * G(z)$ is defined as $H(z) = \sum_{i=0}^{u+v} h_i z^{[i]}$, where $h_i \stackrel{\text{def}}{=} \sum_{j=0}^{i} f_j g_{i-j}^{[j]}$ for $0 \leq i \leq u+v$. Therefore, $H(z)$ can be computed with uv multiplications and $uv-(u+v+1)$ additions. In the case where $u+v \geq m$, we may want to compute $H'(z) = \sum_{i=0}^{m-1} h'_i z^{[i]}$ such that $h'_i = h_i + h_{i+m}$ for $0 \leq i \leq u+v-m$. We hence have $H'(\alpha) = H(\alpha)$ for all $\alpha \in \mathrm{GF}(q^m)$ and $\deg H'(z) < m$. Note that this reduction costs $u+v-m+1$ additions. However, the decoding of Gabidulin codes only considers the case where $u+v < m$, and hence this reduction is unnecessary.

The left-long division of $F(z)$ by $G(z)$ is defined as $F(z) = Q(z) * G(z) + R(z)$, where $\deg R(z) < \deg G(z)$. It can be computed as follows. Set $i = 0$ and $F^{(0)}(z) = F(z)$. While $d_i \stackrel{\text{def}}{=} \deg F^{(i)}(z) \geq \deg G(z)$, calculate

$$Q^{(i)}(z) = \frac{f_{d_i}^{(i)}}{g_v^{[d_i-v]}} z^{[d_i-v]}, \tag{3}$$

$$F^{(i+1)}(z) = F^{(i)}(z) - Q^{(i)}(z) * G(z), \tag{4}$$

and increment i by 1. Return $R(z) = F^{(i)}(z)$ and $Q(z) = \sum_{j=0}^{i-1} Q^{(j)}(z)$.

By (3), determining $Q^{(i)}(z)$ requires 1 inversion and 1 multiplication. However, the inversion of g_v can be computed once at iteration $i = 0$ and stored for the following iterations. By (4), calculating $F^{(i+1)}(z)$ can be done using the multiplication of two LP's, and the addition of two LP's. However, the product $Q^{(i)} * G(z)$ can be implemented with only $v + 1$ multiplications, and we have $f_j^{(i+1)} = f_j^{(i)}$ for $0 \le j \le d_i - v - 1$. Also, we know that $\deg F^{(i+1)}(z) < d_i$, hence the calculation of $f_{d_i}^{(i+1)}$ can be omitted, which further saves 1 multiplication and 1 addition. Therefore, $F^{(i+1)}(z)$ can be computed using v multiplications and v additions. Note that $\deg F^{(i+1)}(z) < \deg F^{(i)}(z)$, and hence the loop will terminate after at most $u - v + 1$ iterations. The worst-case scenario happens when $u - v + 1$ iterations are needed, and hence $d_i = u - i$. We obtain that the complexity of the long division is upper bounded by 1 inversion, $(v + 1)(u - v + 1)$ multiplications, and $v(u - v + 1)$ additions.

IV. Complexities of the extended Euclidean algorithm and the Berlekamp-Massey algorithm

In this section, we determine the complexity of **Step 2** of the decoding algorithm using either the EEA or the BMA. We decide to present the number of operations in terms of m, n, d, and r. We will denote the complexity of a step or LP operation as $[i, m, a]$ henceforth, where i, m, and a denote the numbers of inversions, multiplications, and additions, respectively.

A. Complexity of the extended Euclidean algorithm

The EEA for linearized polynomials proceeds as follows. Let $F_0(z)$ and $F_1(z)$ be two LP's, where $\deg F_1(z) \le \deg F_0(z)$. Then there exists a chain of left-divisions

$$F_i(z) = G_{i+1}(z) * F_{i+1}(z) + F_{i+2}(z), \tag{5}$$

such that $F_{i+2}(z) < \deg F_{i+1}(z)$. These equalities stand for $0 \le i \le s$, where $F_{s+2}(z) = 0$, and the last nonzero remainder $F_{s+1}(z)$ is the right symbolic greatest common divisor of $F_0(z)$ and $F_1(z)$. We introduce the LP's $A_i(z)$ defined as $A_{-1}(z) \stackrel{\text{def}}{=} 0$, $A_0(z) \stackrel{\text{def}}{=} z$ and for $i \ge 1$ by

$$A_i(z) = G_i(z) * A_{i-1}(z) + A_{i-2}(z). \tag{6}$$

The LP's $A_i(z)$ are important in the decoding of Gabidulin codes using the EEA.

The decoding algorithm uses the EEA for $F_0(z) = z^{[d-1]}$ and $F_1(z) = S(z)$. Denoting $d_i \stackrel{\text{def}}{=} \deg F_i(z)$, it can be shown that $d_{i+1} < d_i$, $d_i \le d-1-i$, and $d_r = r$. The EEA may stop after obtaining $A_r(z)$. Note that $\Lambda(z) = \gamma A_r(z)$ for some $\gamma \in \mathrm{GF}(q^m)$. Therefore, finding the roots of $A_r(z)$ is equivalent to finding the roots of $\Lambda(z)$. $A_r(z)$ is obtained after r iterations.

By (5), calculating $F_{i+2}(z)$ takes a long division of LP's. Using our results in Section III, this takes $[1, (d_{i+1} + 1)(d_i - d_{i+1} + 1), d_{i+1}(d_i - d_{i+1} + 1)]$. Summing for i from 0 to $r - 1$, we obtain that the number of inversions to determine $F_2(z), \ldots, F_{r+1}(z)$ is r, while the number of multiplications satisfies

$$\begin{aligned} & \sum_{i=0}^{r-1} (d_{i+1} + 1)(d_i - d_{i+1} + 1) \cdots \\ = & \sum_{i=0}^{r-1} d_{i+1}(d_i - d_{i+1}) + \sum_{i=0}^{r-1} d_i + r \\ \le & (d-2)(d-1-r) + r(d-1) - \frac{1}{2} r(r-1) + r \\ = & (d-1)(d-2) - \frac{1}{2} r(r-5). \end{aligned}$$

Similarly, the number of additions can be upper bounded by $(d-1)(d-2) - \frac{1}{2} r(r-1)$.

By (6), calculating $A_i(z)$ takes a multiplication of LP's and an addition of LP's. Using our results in Section III, we see that the complexity of computing $A_i(z)$ is $[0, (d_{i-1} - d_i)\delta_i, (d_{i-1} - d_i)\delta_i]$, with $\delta_i = \deg A_i(z) \le r$. Therefore, the complexity of obtaining $A_1(z), \ldots, A_r(z)$ is upper bounded by $[0, r(d-1-r), r(d-1-r)]$.

Thus, the complexity of **Step 2** using the EEA is $\left[r, (d-1)(d-2) + \frac{1}{2} r(2d-3r+3), (d-1)(d-2) + \frac{1}{2} r(2d-3r-1)\right]$ over $\mathrm{GF}(q^m)$.

B. Complexity of the Berlekamp-Massey algorithm

The BMA for linearized polynomials proceeds as follows [17], [18]. First, set $L = 0$, $\Lambda^{(0)}(z) = z$, and $B^{(0)}(z) = z$. Then, for $i = 0$ to $d - 2$, repeat the following.

- Step 1: Calculate the discrepancy $\Delta = s_i + \sum_{j=1}^{L} \Lambda_j^{(i)} s_{i-j}^{[j]}$.
- Step 2: If $\Delta = 0$, set $\Lambda^{(i+1)}(z) = \Lambda^{(i)}(z)$ and $B^{(i+1)}(z) = z^{[1]} * B^{(i)}(z)$ and return.
- Step 3: If $\Delta \ne 0$, $\Lambda^{(i+1)}(z) = \Lambda^{(i)}(z) - \Delta z^{[1]} * B^{(i)}(z)$.
- Step 4: If $2L > i$, set $B^{(i+1)}(z) = z^{[1]} * B^{(i)}(z)$ and return.
- Step 5: If $2L \le i$, set $B^{(i+1)}(z) = \Delta^{-1}\Lambda^{(i)}(z)$ and $L = i + 1 - L$.

Step 1 takes L additions and L multiplications. Step 2 only takes cyclic shifts, and its complexity is hence neglected. Step 3 takes $\deg B^{(i)}(z)$ multiplications and $\deg \Lambda^{(i)}(z)$ additions. The complexity of Step 4 can also be neglected. Step 5 takes 1 inversion and $\deg \Lambda^{(i)}(z)$ multiplications.

Note that $\Lambda(z) = \Lambda^{(d-1)}(z)$ is the only outcome of the algorithm necessary for the decoding algorithm. Therefore, the BMA can be terminated after Step 3 of the last iteration, which may save an inversion. Suppose that at iteration i we have $2L \le i$, then Step 5 is reached and L is updated to $L' = i+1-L$. At iteration $i' = i+1$, we thus have $2L' = 2i'-2L \ge 2i' - i' + 1 > i'$, and Step 4 is reached instead. Therefore, Step 5 can only be reached every other iteration, and at most $\lfloor \frac{d-2}{2} \rfloor$ inversions are computed in the algorithm. The degrees of $\Lambda^{(r)}(z)$ and $B^{(r)}$ and the parameter L are always upper bounded by $r + 1$. Therefore, the complexity of the algorithm is at most $\lfloor \frac{d-2}{2} \rfloor$ inversions, $(d-1)(d-2)$ multiplications, and $\frac{1}{2}(d-1)(d-2)$ additions.

Thus, the complexity of **Step 2** using the BMA is $\left[\frac{d-2}{2}, (d-1)(d-2), \frac{1}{2}(d-1)(d-2)\right]$ over $\mathrm{GF}(q^m)$.

V. Complexity of decoding Gabidulin codes

We now analyze the complexity of the rest of the decoding algorithm.

Step 1 involves the multiplication of an n-dimensional vector by an $n \times (d-1)$ matrix. This requires $n(d-1)$ multiplications and $(n-1)(d-1)$ additions. Due to the form of the $\mathbf{H}$ matrix, we can save memory by only storing $\mathbf{h} = (h_0, h_1, \ldots, h_{n-1})$ instead of the entire matrix. This way, some cyclic shifts must also be performed. The memory requirement can be further reduced when $h_0, h_1, \ldots, h_{n-1}$ are part of the normal basis B. Indeed, we have $h_i = h_0^{[i]}$ for $0 \leq i \leq n-1$, hence h_0 suffices to characterize the whole $\mathbf{H}$ matrix. Thus, the complexity of **Step 1** is $[0, n(d-1), (n-1)(d-1)]$ over $\mathrm{GF}(q^m)$.

For **Step 3**, Berlekamp [23] suggested several techniques for finding the roots of $\Lambda(z)$ (or equivalently, those of $A_r(z)$). The first technique is to consider the LP as a polynomial, and to do Chien search [24] (making sure that we only consider linearly independent roots). The second technique is to consider the LP as a linear operator from $\mathrm{GF}(q)^m$ to itself. The problem reduces to finding a basis for the kernel of this operator.

Skachek and Roth [20] gave a probabilistic algorithm for finding roots of an LP, which reduces the problem into finding a basis for the image space of another LP. Such basis can be found by evaluating that polynomial for randomly chosen elements of $\mathrm{GF}(q^m)$. The algorithm can be summarized as follows. Let $F(z)$ be an LP of degree u over $\mathrm{GF}(q^m)$, whose roots form a linear subspace of dimension $s \leq u$.

- Step 1: Determine a linearized polynomial $G(z)$ of degree s which has the same roots as $F(z)$.
- Step 2: Compute $H(z)$ such that $x^{[m]} - x = G(z) * H(z)$.
- Step 3: Set $j = 0$. Until $j = s-1$, randomly select $z_j \in \mathrm{GF}(q^m)$ and calculate $H(z_j)$. If $H(z_0), H(z_1), \ldots, H(z_j)$ are linearly independent, update $j = j+1$.
- Step 4: Return $H(z_0), H(z_1), \ldots, H(z_{r-1})$.

Note that the decoding algorithm for Gabidulin codes only considers linearized polynomials satisfying $\deg F(z) = s$. Therefore, $G(z) = F(z)$ and Step 1 can be omitted. Step 2 is a long division of linearized polynomials as described in Section III, which can be done in $[1, (r+1)(m-r+1), r(m-r+1)]$. We now consider the complexity of Step 3. At iteration j, calculating $H(z_j)$ can be done in $[0, m-r+1, m-r]$. Checking that $H(z_j)$ is linearly independent from $H(z_0), H(z_1), \ldots, H(z_{j-1})$ can be done using Gaussian elimination in $(1, jm, jm)$ over $\mathrm{GF}(q)$. It was shown in [20] that the expected number of field elements z_j to be evaluated is given by $(1-q^{j-r})^{-1}$. It follows that the average complexity of Step 3 is upper bounded by $[0, (r+2)(m-r+1), (r+2)(m-r)]$ over $\mathrm{GF}(q^m)$ and $[r+2, mr(r-1), mr(r-1)]$ over $\mathrm{GF}(q)$.

Thus, the complexity of **Step 3** is $\left[1, (2r+3)(m-r+1), (2r+2)(m-r)+r\right]$ over $\mathrm{GF}(q^m)$ and $[r+2, mr(r-1), mr(r-1)]$ over $\mathrm{GF}(q)$.

Step 4 can be done in two different ways. The first way is to modify the problem into the following system of equations

$$\sum_{j=0}^{r-1} E_j^{[-p]} x_p = s_p^{[-p]} \quad \text{for } 0 \leq p \leq r-1. \tag{7}$$

Hence (7) is a system of r linear equations for the r unknowns x_p. It can hence be solved using Gaussian elimination in $O(r^3)$ operations over $\mathrm{GF}(q^m)$.

The second (and more efficient) way proceeds as follows [2]. We first need to compute the $r \times r$ matrices $\mathbf{A}$ and $\mathbf{Q}$ over $\mathrm{GF}(q^m)$ defined as:

$$\begin{aligned} A_{0,j} &= E_j && (8)\\ A_{i,j} &= 0 \quad \text{for } j < i && (9)\\ &= A_{i-1,j} - \left(\frac{A_{i-1,j}}{A_{i-1,i-1}}\right)^{[-1]} A_{i-1,i-1} \cdots \\ &\quad \text{for } 1 \leq i \leq j \leq r-1, && (10) \end{aligned}$$

and

$$\begin{aligned} Q_{0,p} &= s_p && (11)\\ Q_{i,p} &= 0 \quad \text{for } p > r-1-i,\ i \geq 1 && (12)\\ &= Q_{i-1,p} - \left(\frac{Q_{i-1,p+1}}{A_{i-1,i-1}}\right)^{[-1]} A_{i-1,i-1} \cdots \\ &\quad \text{for } p \leq r-1-i,\ i \geq 1. && (13) \end{aligned}$$

By (8) and (11), the values of $A_{0,j}$ and $Q_{0,p}$ do not require any computations. We now consider the complexity of calculating $A_{i,j}$ and $Q_{i,p}$ in (10) and (13), respectively. We first compute the terms $A_{i-1,i-1}^{-1}$ and $\frac{A_{i-1,i-1}}{(A_{i-1,i-1})^{[-1]}}$ for $i \geq 1$. This requires $r-1$ inversions and multiplications. Then each nontrivial computation of $A_{i,j}$ or $Q_{i,p}$ reduces to 1 multiplication and 1 addition. There are $r(r-1)$ such computations. Therefore, computing $\mathbf{A}$ and $\mathbf{Q}$ requires $r-1$ inversions, $(r+1)(r-1)$ multiplications, and $r(r-1)$ additions.

Finally, we compute $\mathbf{x}$ using

$$\begin{aligned} x_{r-1} &= \frac{Q_{r-1,0}}{A_{r-1,r-1}} && (14)\\ x_i &= A_{i,i}^{-1}\left[Q_{i,0} - \sum_{j=i+1}^{r-1} A_{i,j} x_j\right] \cdots \\ &\quad \text{for } 0 \leq i \leq r-2. && (15) \end{aligned}$$

By (14), the calculation of x_{r-1} takes 1 inversion and 1 multiplication. By (15), the calculation of x_i requires $(r-i)$ multiplications and $(r-1-i)$ additions for $0 \leq i \leq r-2$. Note that the inversion of $A_{i,i}$ was already computed during the calculation of the matrix $\mathbf{A}$. Therefore, computing $\mathbf{x}$ requires 1 inversion, $\frac{1}{2}r(r+1)$ multiplications, and $\frac{1}{2}r(r-1)$ additions.

Hence, the complexity of **Step 4** is $\left[r, (r+1)(3\frac{r}{2}-1), \frac{3}{2}r(r-1)\right]$ over $\mathrm{GF}(q^m)$.

For **Step 5**, we have $\mathbf{x} = \mathbf{h}\mathbf{Y}^T$. Let us denote the expansion of the coordinates of $\mathbf{x}$ and $\mathbf{h}$ with respect to the normal basis B as $\bar{\mathbf{X}} \in \mathrm{GF}(q)^{m\times r}$ and $\bar{\mathbf{H}} \in \mathrm{GF}(q)^{m\times n}$, respectively. We obtain $\bar{\mathbf{X}} = \bar{\mathbf{H}}\mathbf{Y}^T$, or equivalently $\mathbf{Y}^T = \bar{\mathbf{H}}^{-L}\bar{\mathbf{X}}$, where $\bar{\mathbf{H}}^{-L} \in \mathrm{GF}(q)^{n\times m}$ such that $\bar{\mathbf{H}}^{-L}\bar{\mathbf{H}} = \mathbf{I}_n$. The $\bar{\mathbf{H}}^{-L}$ matrix can be pre-computed, so this step only requires a matrix multiplication, i.e. mnr multiplications and additions over $\mathrm{GF}(q)$. Note that if $h_0, h_1, \ldots, h_{n-1}$ are part of the normal basis B, then $\bar{\mathbf{H}} = (\mathbf{I}_n \,|\, \mathbf{0}_{n,m-n})^T$ and hence $\bar{\mathbf{H}}^{-L} = (\mathbf{I}_n \,|\, \mathbf{0}_{n,m-n})$. The multiplication $\bar{\mathbf{H}}^{-L}\bar{\mathbf{X}}$ reduces to selecting the first n rows of $\bar{\mathbf{X}}$. This subclass of Gabidulin codes hence saves computation and memory requirement. Thus, the complexity of **Step 5** is $[0, mnr, mnr]$ over $\mathrm{GF}(q)$ for the general case, $[0, 0, 0]$ if $h_0, h_1, \ldots, h_{n-1}$ are part of the normal basis B.

Step 6 This step is equivalent to the matrix multiplication $\mathbf{E}\mathbf{Y}$, where $\mathbf{E} \in \mathrm{GF}(q)^{m\times r}$ is the expansion of the vector $(E_0, E_1, \ldots, E_{r-1})$ over the normal basis B. This can hence be calculated in mnr multiplications and additions over $\mathrm{GF}(q)$. Thus, the complexity of **Step 6** is $[0, mnr, mnr]$ over $\mathrm{GF}(q)$.

VI. Comments and comparison

We comment on the complexity of the decoding algorithm considered above. The complexities of the EEA and the BMA are close, with the BMA being more efficient. Furthermore, using the subclass of Gabidulin codes for which the parity-check matrix is generated by elements of a normal basis saves $O(mnr)$ operations over $\mathrm{GF}(q)$ and $O(mnd)$ symbols of memory.

From our results in Section V, we conclude that the overall complexity of the decoding algorithm is dominated by the syndrome computation in **Step 1** and finding the roots of $\Lambda(z)$ in **Step 3**, and is on the order of $O(mr)$ operations over $\mathrm{GF}(q^m)$. More precisely, we consider the case where $m = bn$ with $b \geq 1$, $k = Rn = n - d + 1$ with the code rate satisfying $0 < R < 1$, and the error rank satisfying $r = \lfloor \frac{d-1}{2} \rfloor$. As the parameters increase, we have $d \sim n(1-R)$ and $r \sim \frac{1}{2}n(1-R)$. The asymptotic complexity of each step is

- **Step 1** $\left[0, n^2(1-R), n^2(1-R)\right]$,
- **Step 2** $\left[\frac{1}{2}n(1-R), \frac{3}{2}n^2(1-R)^2, \frac{9}{8}n^2(1-R)^2\right]$ for the EEA or $\left[\frac{1}{2}n(1-R), n^2(1-R)^2, \frac{1}{2}n^2(1-R)^2\right]$ for the BMA,
- **Step 3** $\left[1, n^2(1-R)(b-\frac{1}{2}(1-R)), n^2(1-R)(b-\frac{1}{2}(1-R))\right]$,
- **Step 4** $\left[\frac{1}{2}n(1-R), \frac{3}{8}n^2(1-R), \frac{3}{8}n^2(1-R)\right]$,

while the complexities of **Step 5** and **Step 6** are negligible. The overall complexity of the decoding algorithm using the BMA is hence approximated by $\left[n(1-R), \frac{1}{8}n^2(1-R)(15+8b-7R), \frac{1}{8}n^2(1-R)(11+7b-3R)\right]$.

We now compare the complexities of some known decoding algorithms for Gabidulin codes. The PGZ algorithm has a complexity on the order of $O(r^3)$ operations in $\mathrm{GF}(q^m)$ [3]. The WB algorithm was shown in [19] to use $\frac{1}{2}(5n^2-3k^2+n-k)$ multiplications over $\mathrm{GF}(q^m)$. The asymptotic complexity of the WB is hence approximated by $\frac{1}{2}n^2(5-3R^2)$. Therefore, the BMA is more efficient for high rates, while the WB algorithm is more suitable for low rate codes.

References

[1] P. Delsarte, "Bilinear forms over a finite field, with applications to coding theory," *Journal of Combinatorial Theory A*, vol. 25, pp. 226–241, 1978.

[2] E. M. Gabidulin, "Theory of codes with maximum rank distance," *Problems on Information Transmission*, vol. 21, no. 1, pp. 1–12, Jan. 1985.

[3] R. M. Roth, "Maximum-rank array codes and their application to crisscross error correction," *IEEE Trans. Info. Theory*, vol. 37, no. 2, pp. 328–336, March 1991.

[4] E. M. Gabidulin, A. V. Paramonov, and O. V. Tretjakov, "Ideals over a non-commutative ring and their application in cryptology," *LNCS*, vol. 573, pp. 482–489, 1991.

[5] P. Lusina, E. M. Gabidulin, and M. Bossert, "Maximum rank distance codes as space-time codes," *IEEE Trans. Info. Theory*, vol. 49, pp. 2757–2760, Oct. 2003.

[6] R. Koetter and F. R. Kschischang, "Coding for errors and erasures in random network coding," *submitted to IEEE Trans. Info. Theory*, available at http://arxiv.org/abs/cs/0703061.

[7] D. Silva, F. R. Kschischang, and R. Koetter, "A rank-metric approach to error control in random network coding," *submitted to IEEE Trans. Info. Theory*, available at http://arxiv.org/abs/0711.0708.

[8] E. M. Gabidulin, "Optimal codes correcting lattice-pattern errors," *Problems on Information Transmission*, vol. 21, no. 2, pp. 3–11, 1985.

[9] R. M. Roth, "Probabilistic crisscross error correction," *IEEE Trans. Info. Theory*, vol. 43, no. 5, pp. 1425–1438, Sept. 1997.

[10] A. Kshevetskiy and E. M. Gabidulin, "The new construction of rank codes," *Proc. IEEE Int. Symp. on Information Theory*, pp. 2105–2108, Sept. 2005.

[11] E. M. Gabidulin and P. Loidreau, "Properties of subspace subcodes of optimum codes in rank metric," available at http://arxiv.org/pdf/cs.IT/0607108.

[12] M. Gadouleau and Z. Yan, "Error performance analysis of maximum rank distance codes," *Submitted to IEEE Transactions on Information Theory*, available at http://arxiv.org/pdf/cs.IT/0612051.

[13] O. Ore, "On a special class of polynomials," *Transactions of the American Mathematical Society*, vol. 35, pp. 559–584, 1933.

[14] ——, "Contribution to the theory of finite fields," *Transactions of the American Mathematical Society*, vol. 36, pp. 243–274, 1934.

[15] E. Berlekamp, "Nonbinary BCH decoding," *Proc. IEEE Int. Symp. on Info. Theory*, 1967.

[16] J. L. Massey, "Shift register synthesis and BCH decoding," *IEEE Trans. Info. Theory*, vol. 15, no. 1, pp. 122–127, Jan. 1969.

[17] G. Richter and S. Plass, "Fast decoding of rank-codes with rank errors and column erasures," *Proceedings of IEEE ISIT 2004*, p. 398, June 2004.

[18] ——, "Error and erasure of rank-codes with a modified Berlekamp-Massey algorithm," *Proceedings of ITG Conference on Source and Channel Coding 2004*, pp. 203–2112, January 2004.

[19] P. Loidreau, "A Welch-Berlekamp like algorithm for decoding Gabidulin codes," *Proceedings of the 4th International Workshop on Coding and Cryptography*, 2005.

[20] V. Skachek and R. M. Roth, "Probabilistic algorithm for finding roots of linearized polynomials," *to appear in Designs, Codes and Cryptography*, available at http://csi.ucd.ie/~vitalys/Papers/Roots-linearized/linearized-poly-dcc.pdf.

[21] F. MacWilliams and N. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam: North-Holland, 1977.

[22] R. Lidl and H. Niederreiter, *Finite Fields*, ser. Encyclopedia of Mathematics and its Applications, G. Rota, Ed., 1983, vol. 20.

[23] E. Berlekamp, *Algebraic Coding Theory*. Aegean Park Press, 1984.

[24] R. T. Chien, "Cyclic decoding procedure for the Bose-Chaudhuri-Hocquenghem codes," *IEEE Trans. Info. Theory*, vol. 10, pp. 357–363, Oct. 1964.