Algebraic Soft Decoding Algorithms for Reed-Solomon Codes Using Module

Jiongyue Xing, Student Member, IEEE, Li Chen, Senior Member, IEEE,
Martin Bossert, Fellow, IEEE

Abstract

The interpolation based algebraic decoding for Reed-Solomon (RS) codes can correct errors beyond half of the code’s minimum Hamming distance. Using soft information, algebraic soft decoding (ASD) can further enhance the performance. This paper introduces two ASD algorithms that utilize a new interpolation technique, the module minimization (MM). Achieving the desirable Gröbner basis, the MM interpolation is simpler than the existing Koetter’s interpolation that is an iterative polynomial construction process. We will demonstrate how the MM technique can be utilized to solve the interpolation problem in the Koetter-Vardy decoding and the algebraic Chase decoding, respectively. Their re-encoding transformed variants will also be introduced. This transform reduces the size of module entries, leading to a reduced MM complexity. It is more effective for high rate codes. For an RS code of length $n$ and dimension $k$, the MM complexity is $O(n(n - k)l^5)$, where $l$ is the $y$-degree of the interpolated polynomial. Re-encoding transform further reduces the MM complexity to $O((n - k)^2l^5)$. The MM complexity characterizes complexity of the two ASD algorithms. Simulation results will be further provided to substantiate the proposals’ decoding and complexity performances.

Index Terms

Algebraic Chase decoding, complexity, interpolation, Koetter-Vardy decoding, module minimization, Reed-Solomon codes

Jiongyue Xing and Li Chen are with the School of Electronics and Communication Engineering, both are of Sun Yat-sen University, Guangzhou, P. R. China. Martin Bossert is with the Institute of Communications Engineering, Ulm University, Ulm, Germany. Email: xingjyue@mail2.sysu.edu.cn, chenli55@mail.sysu.edu.cn, martin.bossert@uni-ulm.de. This work is sponsored by the National Natural Science Foundation of China (NSFC) with projects ID 61671486 and 61372079.
I. INTRODUCTION

Reed-Solomon (RS) codes are widely employed in data communications and storage systems, in which the well known Berlekamp-Massey (BM) decoding algorithm \[1\] \[2\] is employed. It is a syndrome based decoding that delivers at most one decoded message candidate. Hence, it is also called the unique decoding. The other RS unique decoding algorithms include the Euclidean algorithm \[3\] and the Welch-Berlekamp (WB) algorithm \[4\]. They have an efficient running time but with a limited error-correction capability. Given an \((n,k)\) RS code, where \(n\) and \(k\) are the length and dimension of the code, respectively, they can correct at most \(\lfloor \frac{n-k}{2} \rfloor\) errors, i.e., half of the code’s minimum Hamming distance. Assisted by soft information to perform the error-erasure decoding, the generalized minimum-distance (GMD) algorithm \[5\] and the modified WB algorithm \[6\] both achieve an improved decoding performance.

In late 90s, Sudan introduced an interpolation based algebraic decoding algorithm \[7\] to correct errors beyond the half distance limit. But this improvement only applies to low rate codes. Guruswami and Sudan later improved it to decode all rate codes up to \(n - \lfloor \sqrt{n(k-1)} \rfloor - 1\) errors \[8\]. It is the so-called Guruswami-Sudan (GS) algorithm. Since this interpolation based decoding delivers a list of message candidates, it is also called list decoding. It consists of two steps, interpolation and factorization, while the former dominates the decoding complexity. Interpolation is often implemented by Koetter’s iterative polynomial construction approach \[9\]. It yields the desirable Gröbner basis from which the minimum candidate is chosen to be factorized to retrieve the intended message. Algebraic soft decoding (ASD) was later introduced by Koetter and Vardy, namely the KV algorithm \[10\]. It transforms soft received information into multiplicity information that defines the following interpolation, outperforming its hard-decision prototype without incurring a large computational cost. The other major ASD algorithm is the algebraic Chase decoding (ACD) \[11\]. Using soft received information, it constructs a number of decoding test-vectors. The test-vector formulation allows the following Koetter’s interpolation to construct the interpolated polynomials in a binary tree growth fashion, eliminating the redundant computation. To also decode beyond the half distance bound, power decoding was introduced by Schmidt \textit{et al.} \[12\]. By powering up each received symbol, a virtual interleaved RS code can be created. It achieves a similar error-correction capability as Sudan’s algorithm for low rate codes using multi-sequence shift-register synthesis, implying it is less complex than the algebraic decoding. Other RS soft decoding approaches include the order statistics decoding.
(OSD) [13], the binary image based decoding [14] [15] and the iterative soft decoding [16] [17] that concatenates the adaptive belief propagation (ABP) algorithm and the BM (or the KV) algorithm.

There exists a number of complexity reduction approaches to facilitate the algebraic decoding algorithms. They include the re-encoding transform [18], Wu’s algorithm [19] and its efficient variant [20]. Meanwhile, the average decoding complexity has been addressed in [21] [22]. The progressive KV decoding was introduced to adjust the decoding computation to the quality of received information [22]. Recently, it has been introduced that the interpolation problem can also be solved from the perspective of Gröbner basis of module [23] [24]. In contrast to Koetter’s interpolation that grows the entries of the Gröbner basis in a point-by-point fashion, one can define a module with polynomials that interpolate all the prescribed points. Row reduction is applied to further reduce the module into the desirable Gröbner basis. This can be realized using the Mulders-Storjohann (MS) algorithm [25], the Alekhnovich algorithm [26] or the Divide-and-Conquer algorithm [27]. Such an interpolation technique is called module minimization (MM). It is simpler than Koetter’s interpolation. The MM interpolation approach has been generalized to Wu’s list decoding algorithm and power decoding in [28] and [29], respectively. An MM based multi-trial GS decoding was introduced in [30]. It is also a progressive RS decoding that gradually enlarges the decoding parameter until a satisfied decoding outcome is produced. Performing the ASD using the new MM interpolation technique starts to appear in literature but they have received light attention. These work include [26] [31] [32] that address the KV interpolation problem. More recently, performing ACD using MM has been presented in [33].

This paper aims to present a more intuitive introduction on how to perform the KV and ACD algorithms using the MM interpolation. From this point onward, they are named the KV-MM algorithm and the ACD-MM algorithm, respectively. Our contributions can be outlined as follows:

• We will explain how to use the MM technique to solve the interpolation problem in the two classical ASD algorithms. This is challenging for solving KV interpolation problem in which the number of interpolation points is not restricted to the codeword length and each point can have a different multiplicity. Besides explaining how to solve the KV interpolation problem, we will also show the termination criterion for the reliability transform so that the interpolated polynomial can inherit a designed y-degree (denoted as \(l\)), which is the key decoding parameter for both the KV-MM and the ACD-MM algorithms.
• We will further introduce the re-encoding transformed variants of the KV-MM and the ACD-MM algorithms. The transform helps reduce the degree of module entries leading to a simpler MM interpolation.

• Complexity of the KV-MM and the ACD-MM algorithms are analyzed. Our analysis shows without the re-encoding transform, the MM complexity is $O(n(n-k)l^5)$. Re-encoding transform further reduces it to $O((n-k)^2l^5)$. They both define the complexity of the two ASD algorithms. Therefore, this analysis reveals that the re-encoding transform is more effective for high rate codes, which is of practical interest.

• Finally, simulation results on complexity and decoding performance of the two algorithms will be provided. Our results will show the MM interpolation is far less complex than Koetter’s interpolation and the re-encoding transform can significantly reduce the interpolation complexity. The KV-MM and the ACD-MM algorithms will also be compared under different benchmarks, providing more insights of their applications.

The rest of the paper is organized as follows. Section II introduces RS codes and the GS decoding using MM. Section III introduces the KV-MM algorithm and its re-encoding transformed variant. Section IV introduces the ACD-MM algorithm and its re-encoding transformed variant. Section V analyzes complexity of the above algorithms and Section VI shows our simulation results. Finally, Section VII concludes the paper.

II. RS Codes and Its Algebraic Decoding

This section introduces the prerequisites of the paper, including RS encoding and the GS decoding utilizing the MM interpolation.

A. RS Codes

Let $\mathbb{F}_q = \{\sigma_0, \sigma_1, \ldots, \sigma_{q-1}\}$ denote the finite field of size $q$, and $\mathbb{F}_q[x]$ and $\mathbb{F}_q[x, y]$ denote the univariate and the bivariate polynomial rings defined over $\mathbb{F}_q$, respectively. For an $(n, k)$ RS code, where $n = q - 1$, the message polynomial $f(x) \in \mathbb{F}_q[x]$ is

$$f(x) = f_0 + f_1x + \cdots + f_{k-1}x^{k-1},$$

where $f_0, f_1, \ldots, f_{k-1}$ are message symbols. The codeword $\mathcal{C} = (c_0, c_1, \ldots, c_{n-1}) \in \mathbb{F}_q^n$ can be generated by

$$\mathcal{C} = (f(\alpha_0), f(\alpha_1), \ldots, f(\alpha_{n-1})), \quad (2)$$

where $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ are $n$ distinct nonzero elements of $\mathbb{F}_q$. They are called the code locators.
B. GS Decoding Using MM

Let \( \omega = (\omega_0, \omega_1, \ldots, \omega_{n-1}) \in F_q^n \) denote the received word. The GS decoding algorithm \([8]\) consists of two steps, interpolation and factorization. To perform interpolation, the following \( n \) points are formed

\[
(\alpha_0, \omega_0), (\alpha_1, \omega_1), \ldots, (\alpha_{n-1}, \omega_{n-1}).
\]  

The Hamming distance between \( c \) and \( \omega \) is defined as

\[
d_H(c, \omega) = |\{ j \mid c_j \neq \omega_j, \forall j \}|.
\]  

Given a polynomial \( Q(x, y) = \sum_{a, b} Q_{ab} x^a y^b \in F_q[x, y] \), its \((\mu, \nu)\)-weighted degree is defined as

\[
\deg_{\mu, \nu} Q(x, y) = \max \{ \mu a + \nu b \mid Q_{ab} \neq 0 \}.
\]  

Armed with the above knowledge, the following algebraic decoding theorem can be introduced.

**Theorem 1** \([8]\). For an \((n, k)\) RS code, let \( Q \in F_q[x, y] \) denote the polynomial that interpolates the \( n \) points of (3) with a multiplicity of \( m \). If \( m(n - d_H(c, \omega)) > \deg_{1, k-1} Q(x, y) \), then \( Q(x, f(x)) = 0 \).

Therefore, interpolation finds polynomial \( Q \) with the minimum \((1, k-1)\)-weighted degree, while factorization \([34]\) further finds its \( y \)-roots which contain the intended message polynomial. Hence, the maximum factorization output list size (OLS) is determined by \( \deg_y Q \). This is often used as an algebraic decoding parameter. In this paper, we use \( l \) to denote the maximum OLS and the ASD algorithms are parameterized by \( l \). Note that for the GS decoding, it holds that \( m \leq l \) \([8]\) \([27]\).

Now, we introduce the MM technique for solving the interpolation problem.

**Definition I.** Let \( \xi = (\xi_0(x), \xi_1(x), \ldots, \xi_i(x)) \) denote a vector over \( F_q[x] \), we define the degree of \( \xi \) as

\[
\deg \xi = \max \{ \deg \xi_i(x), \forall i \}.
\]  

The leading position (LP) of vector \( \xi \) is defined as

\[
\text{LP}(\xi) = \max \{ i \mid \deg \xi_i(x) = \deg \xi \},
\]  

and the leading term (LT) of \( \xi \) will be

\[
\text{LT}(\xi) = \xi_{\text{LP}(\xi)}(x).
\]
Since $\xi_t(x) = \xi_t^{(0)} + \xi_t^{(1)} x + \cdots + \xi_t^{(\deg \xi_t(x))} x^{\deg \xi_t(x)}$, the leading coefficient (LC) of $\xi_t(x)$ is

$$\text{LC}(\xi_t(x)) = \xi_t^{(\deg \xi_t(x))}. \quad (9)$$

**Definition II.** Given a matrix $V$ over $\mathbb{F}_q[x]$, we denote its row-$t$ by $V|_t$ and its entry of row-$t$ column-$\tau$ by $V|_t^{(\tau)}$. Furthermore, the degree of $V$ is defined as

$$\deg V = \sum_t \deg V|_t. \quad (10)$$

For GS decoding with an interpolation multiplicity of $m$, which delivers at most $l$ message candidates, a module $M_l$ will be required. It is defined as follows.

**Definition III.** Module $M_l$ is the space of all bivariate polynomials over $\mathbb{F}_q[x, y]$ that interpolate the $n$ points with a multiplicity of $m$ and have a maximum $y$-degree of $l$.

In order to construct $M_l$, the following two polynomials are needed.

$$G(x) = \prod_{j=0}^{n-1} (x - \alpha_j), \quad (11)$$

$$R(x) = \sum_{j=0}^{n-1} \omega_j \Phi_j(x), \quad (12)$$

where

$$\Phi_j(x) = \prod_{j' = 0, j' \neq j}^{n-1} \frac{x - \alpha_j'}{\alpha_j - \alpha_j'} \quad (13)$$

is the Lagrange basis polynomial. It enables $\Phi_j(\alpha_j) = 1$ and $\Phi_j(\alpha_j') = 0, \forall j' \neq j$. Consequently, $R(\alpha_j) = \omega_j, \forall j$. Armed with the above knowledge, module $M_l$ can be formed by the following $l + 1$ generators [23] [27],

$$P_t(x, y) = G(x)^{m-t}(y - R(x))^t, \text{ if } 0 \leq t \leq m, \quad (14)$$

$$P_t(x, y) = y^{l-m}(y - R(x))^m, \text{ if } m < t \leq l. \quad (15)$$

It can be seen that $P_t(\alpha_j, \omega_j) = 0, \forall (t, j)$, and the total degree of $y - R(x)$ and $G(x)$ is $m$. Hence, polynomials $P_t(x, y)$ interpolate the $n$ points of (3) with a multiplicity of $m$. Their maximum $y$-degree is $l$. Since $P_t(x, y) = \sum_{\tau \leq t} P_t^{(\tau)}(x) y^\tau$, where $P_t^{(\tau)}(x) \in \mathbb{F}_q[x]$, module $M_l$ can be represented as an $(l + 1) \times (l + 1)$ matrix over $\mathbb{F}_q[x]$, where its entry $M_l|_t^{(\tau)} = P_t^{(\tau)}(x)$.

**Definition IV.** Let $D_{\beta,l} = \text{diag}(1, x^\beta, \ldots, x^{l\beta})$ and $\tilde{D}_{\beta,l} = \text{diag}(x^{l\beta}, x^{(l-1)\beta}, \ldots, 1)$ denote the diagonal matrices of size $(l + 1) \times (l + 1)$, where $\beta$ is an integer. In this paper, two types of mapping will be needed for $M_l$

$$\mathcal{A}_l = M_l \cdot D_{\beta,l}, \quad (16)$$
\[ A_t = M_t \cdot \tilde{D}_{\beta,t}. \] (17)

Note that \( A_t \) is no longer a module. Inversely, the demapping of \( A_t \) can be

\[ M_t = A_t \cdot D_{-\beta,t}, \] (18)

\[ M_t = A_t \cdot \tilde{D}_{-\beta,t}. \] (19)

A module will be restored by the demapping.

**Definition V** [25]. A square matrix over \( \mathbb{F}_q[x] \) is in the weak Popov form if the leading position of each row is different.

**Lemma 2.** For a square matrix \( V \) over \( \mathbb{F}_q[x] \), when it is in the weak Popov form, we have \( \deg V = \deg \det V \).

**Proof:** For a square matrix \( V \), if the leading position of each row is different, \( \prod_t LT(V|_t) \) will appear in \( \det V \) and \( \deg \det V = \sum_t \deg V|_t = \deg \prod_t LT(V|_t) \).

In order to illustrate the above Definitions and Lemma, the following example is given.

**Example 1.** Given a \( 3 \times 3 \) matrix over \( \mathbb{F}_2[x] \) as

\[
V = \begin{pmatrix}
1 & 1 + x^2 & x + x^2 \\
x^3 & 1 + x^2 & x + x^2 \\
1 + x & 1 + x^3 & 1 + x^2
\end{pmatrix},
\]

we have \( \deg V|_0 = 2, \deg V|_1 = 3 \) and \( \deg V|_2 = 3 \). Hence, \( \deg V = 8 \). The leading position of each row is \( \text{LP}(V|_0) = 2, \text{LP}(V|_1) = 0 \) and \( \text{LP}(V|_2) = 1 \), respectively. Hence, matrix \( V \) is in the weak Popov form. By elaborating \( \det V \), term \((x + x^2)x^2(1 + x^3)\) has the highest degree among all in the matrix determinant. Therefore, \( \deg \det V = 8 = \deg V \).

For the MM interpolation, after the module \( M_t \) has been formed by (14) (15), the mapping of

\[ A_t = M_t \cdot D_{k-1,t} \] (20)

will be performed so that \( \deg A_t|_t = \deg_{1,k-1} P_t(x,y) \). Afterwards, the MS algorithm [25] that is shown in Algorithm [1] will reduce \( A_t \) into the weak Popov form, denoted as \( A'_t \). The demapping of

\[ M'_t = A'_t \cdot D_{-(k-1),t} \] (21)
will be performed. $\mathcal{M}_l'$ is the desirable Gröbner basis. Hence, the minimum interpolated polynomial $Q$ can be taken from a row of $\mathcal{M}_l'$. Let $A^t_i$, denote the row that has the minimum degree, $Q(x, y) = \sum_{r \leq l} Q^{(r)}(x)y^r$ can be retrieved from $\mathcal{M}_l'|^t$ by

$$Q^{(r)}(x) = \mathcal{M}_l'|_t^{(r)}, \forall r.$$  

**Algorithm 1** Mulders-Storjohann Algorithm  

**Input:** $A_l$;  

**Output:** $A'_l$;  

1. While $A_l$ is not in the weak Popov form do  
2. Find two rows $\xi_a$ and $\xi_b$ in $A_l$ such that $\deg \xi_a \leq \deg \xi_b$ and $\text{LP}(\xi_a) = \text{LP}(\xi_b)$;  
3. Perform $\xi_b \leftarrow \xi_b - \frac{\text{LC}(\text{LP}(\xi_a)(x)) \cdot \xi_a}{\text{LC}(\text{LP}(\xi_b)(x))} \cdot \xi_b$;  
4. End while  

**III. THE KV-MM ALGORITHM**

This section introduces the KV-MM algorithm. It transfers the channel observation into a multiplicity matrix which defines the following MM interpolation. Its re-encoding transformed variant will also be introduced.

**A. From Reliability Matrix to Multiplicity Matrix**

Assume codeword $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$ is transmitted through a memoryless channel and $\mathbf{r} = (r_0, r_1, \ldots, r_{n-1}) \in \mathbb{R}^n$ is the received symbol vector, where $\mathbb{R}$ denotes the channel output alphabet. The channel observation is represented by a reliability matrix $\Pi$ whose entries are the a posteriori probability (APP) defined as

$$\pi_{ij} = \Pr[c_j = \sigma_i \mid r_j], \text{ for } 0 \leq i \leq q - 1 \text{ and } 0 \leq j \leq n - 1.$$  

(23)

Matrix $\Pi$ will be proportionally transformed into a multiplicity matrix $M$ using Algorithm A of [10]. $M$ is also a $q \times n$ matrix. Its entry $m_{ij}$ indicates the interpolation multiplicity for point $(\alpha_j, \sigma_i)$. Therefore, matrix $M$ implies $|\{m_{ij} \mid m_{ij} \neq 0\}|$ interpolation points. Interpolation is to find the minimum polynomial $Q(x, y)$ that interpolates all points $(\alpha_j, \sigma_i)$ with a multiplicity of

$^1$It is defined under the assumption that $\Pr[c_j = \sigma_i] = \frac{1}{q}, \forall i$. 

December 5, 2017 DRAFT
Since the decoding is parameterized by the maximum factorization OLS \( l \) and \( l = \text{deg}_y Q \), the \( \mathbf{\Pi} \rightarrow \mathbf{M} \) transform terminates when \( \text{deg}_y Q \) reaches a predefined value of \( l \). Let us define

\[
m_j = \sum_{i=0}^{q-1} m_{ij}
\]

and

\[
m = \max\{m_j, \forall j\}.
\]

The \( \mathbf{\Pi} \rightarrow \mathbf{M} \) transform terminates when \( m = l \). The following subsection will show how to construct the module \( \mathcal{M}_l \) based on \( \mathbf{M} \). This \( \mathbf{\Pi} \rightarrow \mathbf{M} \) termination criterion can also be explained.

**B. Module Formulation and Minimization**

For the KV-MM algorithm, we need to construct the module \( \mathcal{M}_l \) whose bivariate polynomials interpolate points \((\alpha_j, \sigma_i)\) with a multiplicity of at least \( m_{ij} \) and have a maximum \( y \)-degree of \( l \). In the KV algorithm, interpolation points can have different multiplicities. However, generators \( 14, 15 \) can only form a module that holds the same multiplicity for all the points. They cannot be applied. To construct the desirable module, the following point enumeration is needed.

Let \( L_j \) denote a list that enumerates interpolation points \((\alpha_j, \sigma_i)\) from column \( j \) of \( \mathbf{M} \) with their multiplicity \( m_{ij} \) as

\[
L_j = \{ (\alpha_j, \sigma_i), \ldots, (\alpha_j, \sigma_i), \forall i \text{ and } m_{ij} \neq 0 \}.
\]

Note that \(|L_j| = m_j\). Its balanced list \( L_j' \) can be further created as follows. Copy one of the most frequent elements in \( L_j \) to \( L_j' \) and remove it from \( L_j \). Repeat this process \( m_j \) times until \( L_j \) becomes an empty set. Consequently, the balanced list can be denoted as

\[
L_j' = \{ (\alpha_j, y_j^{(0)}), (\alpha_j, y_j^{(1)}), \ldots, (\alpha_j, y_j^{(m_j-1)}) \},
\]

where \( y_j^{(0)}, y_j^{(1)}, \ldots, y_j^{(m_j-1)} \in \mathbb{F}_q \) and they may not be distinct. \((\alpha_j, y_j^{(0)})\) is the first point being introduced into \( L_j' \), followed by \((\alpha_j, y_j^{(1)})\) and etc. The balanced list \( L_j' \) is a permutation of \( L_j \) and therefore \(|L_j'| = m_j\). Further let \( L_j'(s) \) denote a reduced balanced list that contains the last \( m_j - s \) elements of \( L_j' \) as

\[
L_j'(s) = \{ (\alpha_j, y_j^{(s)}), (\alpha_j, y_j^{(s+1)}), \ldots, (\alpha_j, y_j^{(m_j-1)}) \},
\]
where $|L'_j(s)| = m_j - s$. Finally, let $m_j(s)$ denote the maximum multiplicity of elements in $L'_j(s)$ as

$$m_j(s) = \max\{\text{multiplicity}(s_j, y_j^{(\varepsilon)}) \mid \varepsilon = s, s + 1, \ldots, m_j - 1\}. \tag{29}$$

Note that $m_j(0) = \max\{m_{ij}, \forall i\}$ and $m_j(0) = 0$ for $\varepsilon \geq m_j$.

The following example illustrates the above point enumeration process.

**Example 2.** In decoding a $(7, 3)$ RS code, the multiplicity matrix is obtained as Fig. 1 (a). The enumeration lists $L_0 \sim L_6$ and their balanced lists $L'_0 \sim L'_6$ are shown in Figs. 1 (b) and 1 (c), respectively. Hence, when $s = 0$, $m_0(0) = 4$, $m_1(0) = 5$, $m_2(0) = 3$, $m_3(0) = 5$, $m_4(0) = 5$, $m_5(0) = 4$ and $m_6(0) = 3$. When $s = 1$, $m_0(1) = 3$, $m_1(1) = 4$, $m_2(1) = 3$, $m_3(1) = 4$, $m_4(1) = 4$, $m_6(1) = 3$ and $m_0(1) = 2$.

With the above point enumeration, module construction of the KV interpolation can be reinterpreted as follows.

**Remark 1.** In the KV interpolation, we need to construct a module $\mathcal{M}_l$ whose bivariate polynomials interpolate all the enumeration points in the balanced lists with a multiplicity of one for each. Their maximum $y$-degree is $l$.

Now, it is sufficient to introduce the construction of $\mathcal{M}_l$ for the KV interpolation. First, let us define

$$F_\varepsilon(x) = \sum_{j=0}^{n-1} y_j^{(\varepsilon)} \Phi_j(x), \tag{30}$$

where $\varepsilon = 0, 1, \ldots, l-1$. Based on (13), we know $F_\varepsilon(\alpha_j) = y_j^{(\varepsilon)}$, $\forall j$. Hence, $y - F_\varepsilon(x)$ interpolates points $(\alpha_j, y_j^{(\varepsilon)})$, $\forall j$. Note that in formulating the $l$ polynomials $F_0(x) \sim F_{l-1}(x)$, if $m_j < l$, we can assume $y_j^{(\varepsilon)} = 0$ for $\varepsilon \geq m_j$. Furthermore, the module generators can be defined as

$$P_\varepsilon(x, y) = \prod_{j=0}^{n-1} (x - \alpha_j)^{m_j(l)} \prod_{\varepsilon=0}^{l-1} (y - F_\varepsilon(x)), \tag{31}$$
where \( t = 0,1,\ldots,l \). There are \( l+1 \) module generators and their maximum \( y \)-degree is \( l \). Therefore, the above mentioned \( \Pi \rightarrow M \) transform terminates when \( m = l \). When \( t = s \), \( \prod_{s=0}^{s-1}(y - F_s(x)) \) interpolates points \( \{(\alpha_j, y_j^{(0)}), (\alpha_j, y_j^{(1)}), \ldots, (\alpha_j, y_j^{(s-1)})\} \) with a multiplicity of one. Meanwhile, \( \prod_{j=0}^{n-1}(x - \alpha_j)^{m_j(s)} \) interpolates the remaining points \( \{(\alpha_j, y_j^{(s)}), (\alpha_j, y_j^{(s+1)}), \ldots, (\alpha_j, y_j^{(m_j-1)})\} \) with a multiplicity of \( m_j(s) \). Based on Remark 1, generators of (31) formulate the desirable module for the KV interpolation.

When the interpolation property of point \( (\alpha_j, y_j^{(s)}) \) is contained by polynomial \( y - F_s(x) \), it is desirable to reduce the remaining points’ maximum multiplicity so that generator \( P_{s+1}(x, y) \) inherits a smaller \( x \)-degree. Consequently, complexity of the following MS algorithm can be reduced. The balanced lists \( L_j' \) are created for such purpose.

Presenting module \( \mathcal{M}_t \) as a matrix over \( \mathbb{F}_q[x] \), we can generate \( \mathcal{A}_t \) by the mapping of (20). Algorithm \( \Pi \) will then reduce \( \mathcal{A}_t \) into the weak Popov form \( \mathcal{A}_t' \). De-map it as in (21). The interpolated polynomial \( Q(x, y) \) can be retrieved from \( \mathcal{M}_t' \) as in (22).

C. The Re-encoding Transformed KV-MM

Re-encoding reduces the decoding complexity by transforming the interpolation points. First, we will sort \( m_0(0), m_1(0), \ldots, m_{n-1}(0) \) to obtain an index sequence \( j_0, j_1, \ldots, j_{n-1} \) such that \( m_{j_0}(0) \leq m_{j_1}(0) \leq \cdots \leq m_{j_{n-1}}(0) \). Let us define \( \Upsilon = \{j_{n-k}, j_{n-k+1}, \ldots, j_{n-1}\} \) and \( \bar{\Upsilon} = \{j_0, j_1, \ldots, j_{n-k-1}\} \). The \( k \) points in the form of \( (\alpha_j, y_j^{(0)}), j \in \Upsilon \) are chosen to form the re-encoding polynomial that is defined as

\[
\hat{H}(x) = \sum_{j \in \Upsilon} y_j^{(0)} \hat{h}_j(x),
\]

where

\[
\hat{h}_j(x) = \prod_{j' \in \Upsilon, j' \neq j} \frac{x - \alpha_{j'}}{\alpha_j - \alpha_{j'}}.
\]

Consequently, \( \hat{H}(\alpha_j) = y_j^{(0)}, \forall j \in \Upsilon \). Subsequently, all interpolation points \( (\alpha_j, y_j^{(e)}) \) will be transformed by

\[
(\alpha_j, w_j^{(e)}) = (\alpha_j, y_j^{(e)} - \hat{H}(\alpha_j)).
\]

For \( j \in \Upsilon \), if \( y_j^{(e)} = y_j^{(0)}, w_j^{(e)} = 0 \). We further define \( \Lambda_\varepsilon = \{j \mid w_j^{(e)} = 0, j \in \Upsilon\} \) and \( \bar{\Lambda}_\varepsilon = \Upsilon \setminus \Lambda_\varepsilon \). With the transformed interpolation points \( (\alpha_j, w_j^{(e)}) \), polynomials \( F_\varepsilon(x) \) of (30) can be redefined as

\[
F_\varepsilon(x) = \sum_{j=0}^{n-1} w_j^{(e)} \Phi_j(x).
\]
Now let us define
\[ \phi(x) = \prod_{j \in \Upsilon} (x - \alpha_j)^{m_j(0)} \] (36)
and
\[ \psi(x) = \prod_{j \in \Upsilon} (x - \alpha_j). \] (37)

We can introduce the following Lemma that describes the property of module generators when the re-encoding transform is applied.

**Lemma 3.** Given the multiplicity matrix M and the subsequently transformed interpolation points of (34), we have \( \phi(x) | P_t(x, y\psi(x)) \).

**Proof:** Due to its length, this proof is given in Appendix A.

Based on Lemma 3, the following mapping
\[ \phi(x)^{-1}P_t(x, y\psi(x)) \mapsto \tilde{P}_t(x, y) \] (38)
can be performed on the generators of (31) so that the degree of module entries can be reduced. This leads to a simpler row operation of Algorithm 1. Note that generators \( \tilde{P}_t(x, y) \) will not interpolate points \( (\alpha_j, w_j^{(c)}), \forall j \in \Upsilon \). They only form an isomorphism of the desirable module.

Let us further define
\[ \tilde{w}_j^{(c)} = \frac{w_j^{(c)}}{\prod_{j' = 0, j' \neq j}^n (\alpha_j - \alpha_{j'})} \] (39)
and
\[ T_\varepsilon(x) = \sum_{j \in \Upsilon \cup \Lambda} \tilde{w}_j^{(c)} \prod_{j' \in \Upsilon \cup \Lambda, j' \neq j} (x - \alpha_{j'}) \] (40)
the proof of Lemma 3 reveals
\[ P_t(x, y\psi(x)) = \phi(x) \cdot \prod_{j \in \Upsilon} (x - \alpha_j)^{m_j(t)} \cdot \prod_{j \in \Lambda_t} (x - \alpha_j) \cdot \prod_{\varepsilon = 0}^{t-1} \left( y \prod_{j \in \Lambda_\varepsilon} (x - \alpha_j) - T_\varepsilon(x) \right). \] (41)

Based on (38), generators of the module isomorphism are defined as
\[ \tilde{P}_t(x, y) = \prod_{j \in \Upsilon} (x - \alpha_j)^{m_j(t)} \cdot \prod_{j \in \Lambda_t} (x - \alpha_j) \cdot \prod_{\varepsilon = 0}^{t-1} \left( y \prod_{j \in \Lambda_\varepsilon} (x - \alpha_j) - T_\varepsilon(x) \right) \] (42)
and \( t = 0, 1, \ldots, l \). We denote this isomorphism spanned by \( \tilde{P}_t(x, y) \) as \( \varphi(M_t) \). It can also be presented as a matrix over \( \mathbb{F}_q[x] \).

With the re-encoding transform, polynomials are ordered under the \((1, -1)\)-weighted degree. However, performing \( A_t = \varphi(M_t) \cdot D_{-1,t} \) will cause some of the isomorphism entries leaving
\( \mathbb{F}_q[x] \). Instead, \( A_t \) will be generated by the mapping of (17) with \( \beta = 1 \), so that \( \deg A_t |_{t} = \deg_{1,-1} \tilde{P}_t(x, y) + l \). Algorithm [1] will then reduce \( A_t \) into the weak Popov form \( A'_t \). Demap it to \( \varphi(\mathcal{M}'_t) \) as in (19) and polynomial \( \tilde{Q} \) can be retrieved from \( \varphi(\mathcal{M}'_t) \) as in (22). Finally, the desirable interpolated polynomial \( Q \) can be restored by

\[
Q(x, y) = \phi(x)\tilde{Q}\left(x, \frac{y}{\psi(x)}\right). \tag{43}
\]

Factorization [34] will be performed to determine the \( y \)-roots of \( Q \), yielding candidates \( f'(x) \). The intended message polynomial \( \hat{f}(x) \) can be estimated by

\[
\hat{f}(x) = f'(x) + \hat{H}(x). \tag{44}
\]

IV. THE ACD-MM ALGORITHM

This section introduces the ACD-MM algorithm. Instead of transforming the reliability matrix into a multiplicity matrix, it formulates a number of decoding test-vectors based on which the GS decoding will be performed. We will also introduce its re-encoding transformed variant.

A. From Reliability Matrix to Test-Vectors

With reliability matrix \( \Pi \), let

\[
i^{I}_j = \arg \max \{\pi_{ij}, \forall i\} \quad \text{and} \quad i^{II}_j = \arg \max \{\pi_{ij}, \forall i \text{ and } i \neq i^{I}_j\}, \tag{44}
\]

respectively, where function \( \arg \max \) returns to the desirable index \( i \). The two most likely decisions for symbol \( c_j \) are

\[
r^{I}_j = \sigma^{I}_j \quad \text{and} \quad r^{II}_j = \sigma^{II}_j, \tag{45}
\]

respectively. Define the symbol wise reliability metric as [11]

\[
\gamma_j = \frac{\pi_{j,j}}{\pi_{j,j}}, \tag{46}
\]

where \( \gamma_j \in (0, 1) \). With \( \gamma_j \to 0 \), the decision on \( c_j \) is more reliable, and vise versa. By sorting the \( n \) reliability metrics in an ascending order, we obtain a refreshed symbol index sequence \( j_0, j_1, \ldots, j_{n-1} \). It indicates \( \gamma_{j_0} < \gamma_{j_1} < \cdots < \gamma_{j_{n-1}} \). Choose \( \eta \ (\eta < n) \) least reliable symbols that can be realized as either \( r^{I}_j \) or \( r^{II}_j \). For the remaining \( n - \eta \) reliable symbols, they will be realized as \( r^{I}_j \). We can formulate \( 2^n \) interpolation test-vectors which can be generally written as

\[
\Xi_u = (r^{(u)}_{j_0}, r^{(u)}_{j_1}, \ldots, r^{(u)}_{j_{k-1}}, r^{(u)}_{j_k}, \ldots, r^{(u)}_{j_{n-1}}), \tag{47}
\]

where \( u = 1, 2, \ldots, 2^n \) and

\[
r^{(u)}_{j} = r^{I}_j, \quad \text{if} \quad j = j_0, j_1, \ldots, j_{n-\eta-1}. \tag{48}
\]
\[ r_j^{(u)} = r_j^1 \text{ or } r_j^\Pi, \quad \text{if } j = j_{n-\eta}, j_{n-\eta+1}, \ldots, j_{n-1}. \quad (49) \]

Note that the unreliable symbols can also be considered with other less likely decisions. By doing so, the decoding complexity will increase exponentially. However, our research has shown little performance gain can be achieved by considering more than two decisions for each symbol.

**B. Module Formulation and Minimization**

For each test-vector, the MM based GS decoding that is described in Section II.B will be performed. In particular, given a test-vector \( r_u \), polynomial \( R(x) \) of (12) will be redefined as

\[
R_u(x) = \sum_{j=0}^{n-1} r_j^{(u)} \Phi_j(x). \quad (50)
\]

Consequently, \( R_u(\alpha_j) = r_j^{(u)}, \forall j \). Module \( M_l \) can be formed using the \( l + 1 \) generators of (14) (15), in which \( R(x) \) is replaced by \( R_u(x) \).

Note that for the ACD-MM algorithm, the MM interpolation for each test-vector is independent. They can be performed in parallel, leveraging the decoding latency to that of a single GS decoding event. This is an advantage over the ACD algorithm that employs Koetter’s interpolation [11].

**C. The Re-encoding Transformed ACD-MM**

Re-encoding transforms the test-vectors so that they have at most \( k \) extra zero symbols. This again reduces the degree of the module entries, resulting in a simpler row operation of Algorithm [11]. Let \( \Theta = \{j_0, j_1, \ldots, j_{k-1}\} \) denote the index set of the \( k \) most reliable symbols. They will be chosen for re-encoding. Hence, \( \Theta = \{j_k, j_{k+1}, \ldots, j_{n-1}\} \) denotes the index set of the remaining \( n - k \) symbols. We now restrict \( \eta \leq n - k \) so that the \( 2^n \) test-vectors share at least \( k \) common symbols \( r_{j_0}^1, r_{j_1}^1, \ldots, r_{j_{k-1}}^1 \). The \( k \) re-encoding points are \( (\alpha_j, r_j^1) \) and \( j \in \Theta \). For the ACD-MM algorithm, the re-encoding polynomial is

\[
\tilde{H}(x) = \sum_{j \in \Theta} r_j^1 \tilde{h}_j(x), \quad (51)
\]

where

\[
\tilde{h}_j(x) = \prod_{j' \in \Theta, j' \neq j} \frac{x - \alpha_{j'}}{\alpha_j - \alpha_{j'}}. \quad (52)
\]

Hence, \( \tilde{H}(\alpha_j) = r_j^1, \forall j \in \Theta \). All test-vectors \( r_u \) are transformed by

\[
r_u \mapsto z_u : z_j^{(u)} = r_j^{(u)} - \tilde{H}(\alpha_j), \forall j. \quad (53)
\]
The transformed test-vectors can be generally written as
\[ z_{u} = (0, 0, \ldots, 0, z_{j_{k}}^{(u)}, \ldots, z_{j_{n-1}}^{(u)}). \] (54)

With a transformed test-vector \( z_{u} \), polynomial \( R(x) \) of (12) is redefined as
\[ R_{u}(x) = \sum_{j=0}^{n-1} z_{j}^{(u)} \Phi_{j}(x). \] (55)

With \( z_{j}^{(u)} = 0, \forall j \in \Theta \),
\[ V(x) = \prod_{j \in \Theta} (x - \alpha_{j}) \] (56)
becomes the GCD for both the \( G(x) \) of (11) and the \( R_{u}(x) \) of (55). Therefore, given a test-vector \( z_{u} \), we define
\[ \tilde{G}(x) = \frac{G(x)}{V(x)} = \prod_{j \in \Theta} (x - \alpha_{j}) \] (57)
and
\[ \tilde{R}_{u}(x) = \frac{R_{u}(x)}{V(x)} = \sum_{j \in \Theta} \frac{z_{j}^{(u)}}{\omega_{j}} \prod_{j' \in \bar{\Theta}, j' \neq j} (x - \alpha_{j'}), \] (58)
where \( \omega_{j} = \prod_{j' = 0, j' \neq j}^{n-1} (\alpha_{j} - \alpha_{j'}). \)

The following Lemma further reveals the property of module generators.

**Lemma 4.** Given a test-vector in the form of (54) and a multiplicity of \( m \), we have \( V(x)^{m} \mid P_{t}(x, yV(x)) \).

**Proof:** With the generators defined by (14) and (15), when \( 0 \leq t \leq m \), \( P_{t}(x, yV(x)) \) can be elaborated as
\[ G(x)^{m-t}(-R_{u}(x))^{t} + \binom{t}{1} G(x)^{m-t}(-R_{u}(x))^{t-1}V(x)y + \ldots + G(x)^{m-t}(V(x)y)^{t}. \]
When \( m < t \leq l \), \( P_{t}(x, yV(x)) \) becomes
\[ (-R_{u}(x))^{m}(V(x)y)^{t-m} + \binom{m}{1} (-R_{u}(x))^{m-1}(V(x)y)^{t-m+1} + \ldots + (V(x)y)^{t}. \]
Since \( V(x) \mid G(x) \) and \( V(x) \mid R_{u}(x) \), it can be concluded that \( V(x)^{m} \mid P_{t}(x, yV(x)) \).

Armed with the above Lemma, the following mapping can be performed on module generators of (14) and (15) as
\[ V(x)^{-m} P_{t}(x, yV(x)) \mapsto \tilde{P}_{t}(x, y). \] (59)

It results in
\[ \tilde{P}_{t}(x, y) = \tilde{G}(x)^{m-t}(y - \tilde{R}_{u}(x))^{t}, \text{ if } 0 \leq t \leq m, \] (60)
\( \tilde{P}_t(x, y) = (yV(x))^{t-m}(y - \tilde{R}_u(x))^m, \) if \( m < t \leq l. \) \hspace{1cm} (61)

They are the generators of module isomorphism \( \varphi(M_t) \) for the re-encoding transformed ACD-MM algorithm. Presenting \( \varphi(M_t) \) as a matrix over \( \mathbb{F}_q[x] \), we can perform mapping of (17) with \( \beta = 1 \) so that \( \deg \mathcal{A}_t \mid_t = \deg_{t-1} \tilde{P}_t(x, y) + l \). Algorithm 1 will be applied to reduce \( \mathcal{A}_t \) into the weak Popov form \( \mathcal{A}'_t \). Demap it into \( \varphi(M'_t) \) using (19) and polynomial \( \tilde{Q} \) can be retrieved from \( \varphi(M'_t) \) as in (22). Finally, the desirable interpolated polynomial \( Q \) can be restored by

\[
Q(x, y) = V(x)^m \tilde{Q} \left( x, \frac{y}{V(x)} \right).
\hspace{1cm} (62)
\]

If \( f'(x) \) is the factorization output candidate, the intended message polynomial \( \hat{f}(x) \) can be further obtained by \( \hat{f}(x) = f'(x) + \bar{H}(x) \).

V. COMPLEXITY ANALYSIS

This section analyzes complexity of the above mentioned ASD algorithms. We will also look into the complexity when the re-encoding transform is applied. Our analysis will lead to a parameterized comparison between the KV-MM and the ACD-MM algorithms.

A. Without the Re-encoding Transform

For the above mentioned ASD algorithms, their complexity are dominated by the MM interpolation. In particular, it is the row operation of Algorithm 1.

We first analyze the MM complexity without the re-encoding transform. This is determined by the degree of entry \( \mathcal{A}_t \mid_t^{(r)} \) and the number of row operations that is required to reduce \( \mathcal{A}_t \) into the weak Popov form. The following Theorem characterizes the number of row operations that is needed to reduce \( \mathcal{A}_t \) into the weak Popov form.

**Theorem 5** [27]. Given a matrix \( \mathcal{A}_t \) over \( \mathbb{F}_q[x] \), there are less than \( (l+1)(\deg \mathcal{A}_t - \deg \det \mathcal{A}_t + l) \) row operations to reduce it into the weak Popov form \( \mathcal{A}'_t \).

The following two Lemmas further characterize \( \deg \mathcal{A}_t \mid_t^{(r)} \) and \( \deg \mathcal{A}_t - \deg \det \mathcal{A}_t \), respectively.

**Lemma 6.** Without the re-encoding transform, \( \deg \mathcal{A}_t \mid_t^{(r)} \leq nl. \)

**Proof:** For the KV-MM algorithm, the generators (31) and mapping (20) lead to

\[
\deg \mathcal{A}_t \mid_t^{(r)} \leq n(l - t) + (n - 1)(t - \tau) + (k - 1)\tau \\
= nl - t - (n - k)\tau.
\]
Therefore, \( \max\{\deg A_l^{(r)}, \forall (t, \tau)\} = \deg A_l^{(0)} = nl. \)

For the ACD-MM algorithm, we can determine \( \deg A_l^{(r)} \) based on the generators (14) (15) and mapping (20). When \( 0 \leq t \leq m \),

\[
\deg A_l^{(r)} = n(m - t) + (n - 1)(t - \tau) + (k - 1)\tau
= nm - t - (n-k)\tau.
\]

Hence, \( \max\{\deg A_l^{(r)}, 0 \leq t \leq m\} = \deg A_l^{(0)} = nm. \) When \( m < t \leq l \),

\[
\deg A_l^{(r)} = (n - 1)(t - \tau) + (k - 1)\tau
= (n - 1)t - (n-k)\tau,
\]

and \( \max\{\deg A_l^{(r)}, m < t \leq l\} = \deg A_l^{(0)} = (n-1)l. \) Therefore, for the ACD-MM algorithm,

\[
\max\{\deg A_l^{(r)}, \forall (t, \tau)\} = \max\{nm, (n-1)l\} \leq nl.
\]

**Lemma 7.** Without the re-encoding transform, \( \deg A_l - \deg \det A_l \leq \frac{1}{2}(n-k)(l^2 + l). \)

**Proof:** Due to its length, this proof is given in Appendix B.

Based on Theorem 5 and Lemma 7, we know there are at most \( \frac{1}{2}(n-k)l(l+1)(l+2) \) row operations in the MM process. Since \( \deg A_l^{(r)} \leq nl \) and there are \( l+1 \) entries in each row, the MM process requires at most \( \frac{1}{2}n(n-k)l^2(l+1)^2(l+2) \) finite field operations. This leads us to the following conclusion on the MM complexity.

**Theorem 8.** For both the KV-MM and the ACD-MM algorithms, the MM complexity is \( O(n(n-k)l^5). \)

Since the module formulation and factorization have a complexity of \( O(n) \) and \( O(k^2l^3) \), respectively, the following conclusions on the KV-MM and ACD-MM complexity can be straightforwardly led to.

**Corollary 9.** Complexity of the KV-MM algorithm is \( O(n(n-k)l^5). \)

**Corollary 10.** Complexity of the ACD-MM algorithm is \( O(2^\rho n(n-k)l^5). \)

**B. With the Re-encoding Transform**

Re-encoding transform reduces the degree of module entries leading to a simpler row operation during the MM process. The following Lemma characterizes \( \deg A_l^{(r)} \).

**Lemma 11.** With the re-encoding transform, \( \deg A_l^{(r)} \leq (n-k+1)l. \)

**Proof:** With re-encoding transform, entry size of \( A_l \) in the KV-MM algorithm is

\[
\deg A_l^{(r)} \leq (n-k)(l - t) + (n-k-1)(t - \tau) + (l - \tau).
\]
Therefore, \( \max\{\deg A_l^{(r)}, \forall (t, \tau)\} = \deg A_l^{(0)} = (n - k + 1)l. \)

For the ACD-MM algorithm, when \( 0 \leq t \leq m \),
\[
\deg A_l^{(r)} = (n - k)(m - t) + (n - k - 1)(t - \tau) + (l - \tau).
\]
Since \( m \leq l \), \( \max\{\deg A_l^{(r)}, 0 \leq t \leq m\} = \deg A_l^{(0)} = (n - k + 1)l. \) When \( m < t \leq l \),
\[
\deg A_l^{(r)} = k(t - m) + (n - k - 1)(t - \tau) + (l - \tau),
\]
and \( \max\{\deg A_l^{(r)}, m < t \leq l\} = \deg A_l^{(0)} = (n - k)l. \)

Despite the re-encoding transform can reduce degree of module entries, it does not lead to a significant reduction on the number of row operations. The following Lemma and its proof further reveal this aspect.

**Lemma 12.** With the re-encoding transform, \( \deg A_l - \deg \det A_l \leq \frac{1}{2}(n - k)(l^2 + l). \)

**Proof:** Similar to the proof of Lemma 7, let \( \tau_t \) identify the maximum entry of row \( A_l|_t \).
For the KV-MM algorithm, we have
\[
\deg A_l - \deg \det A_l \leq \sum_{t=0}^{l} ((n - k + 1)(t - \tau_t) + (l - \tau_t) - (l - t)).
\]
Therefore, when \( \tau_t = 0 \), \( \max\{\deg A_l - \deg \det A_l\} = \frac{1}{2}(n - k)(l^2 + l). \)

For the ACD-MM algorithm, we have
\[
\deg A_l = \sum_{t=0}^{m} ((n - k)m - t + l) + \sum_{t=m+1}^{l} ((n - 1)t + (l - km))
\]
and
\[
\deg \det A_l = \sum_{t=0}^{m} ((n - k)(m - t) + (l - t)) + \sum_{t=m+1}^{l} (k(t - m) + (l - t)).
\]
Therefore, \( \deg A_l - \deg \det A_l = \sum_{t=0}^{l}(n - k)t = \frac{1}{2}(n - k)(l^2 + l). \)

Recalling Lemma 7, the above Lemma shows that re-encoding transform does not attribute to reducing the number of row operations during the MM process. Further based on Theorem 5, it is sufficient to reach the following Theorem that describes the MM complexity when the re-encoding transform is applied.

**Theorem 13.** For both the re-encoding transformed KV-MM and ACD-MM algorithms, the MM complexity is \( O((n - k)^2 l^5). \)

It should be pointed out that the re-encoding complexity is \( O(k^2) \). When \( l = 1 \), this outweighs the MM interpolation complexity for high rate codes. Moreover, the complexity reduction brought
by the transform cannot compensate the extra re-encoding computation. However, when the
factorization OLS $l$ increases, the re-encoding complexity becomes marginal in the overall
decoding complexity. Re-encoding becomes beneficial for facilitating the decoding. Assuming
the MM interpolation dominates the decoding complexity, the following two Corollaries further
conclude complexity of the re-encoding transformed KV-MM and ACD-MM algorithms.

**Corollary 14.** Complexity of the re-encoding transformed KV-MM algorithm is $O((n-k)^2l^5)$.

**Corollary 15.** Complexity of the re-encoding transformed ACD-MM algorithm is $O(2^n(n-k)^2l^5)$.

The above analysis shows for both the two ASD algorithms, re-encoding transform reduces
the decoding complexity by a factor of $\frac{k}{n}$. This implies for high rate codes, the complexity
reduction is more effective and this is of practical interest.

C. **Comparison Remark**

With the above analysis, more insights in comparing the KV-MM and the ACD-MM algorithms
can be revealed. We compare them in the benchmarks of the maximum number of decoding
outputs and the decoding complexity. The algebraic decoding algorithms are often parameterized
by their decoding output cardinality. The first comparison benchmark is chosen under this agenda.
On the other hand, with the same decoding complexity, error-correction performance superiority
between these two ASD algorithms is still unknown in literature. The complexity benchmark
aims to provide more insights in this aspect, sustaining their future applications. Simulation
results comparing the two algorithms under these benchmarks will be provided in Section VI.

In order to distinguish the decoding parameter $l$ for the two ASD algorithms, we now use
$l_{\text{KV-MM}}$ and $l_{\text{ACD-MM}}$ to denote the maximum factorization OLS in the KV-MM algorithm and
the ACD-MM algorithm, respectively. In particular, $l_{\text{ACD-MM}}$ denotes maximum OLS of a single
GS decoding event. They both define the module size. If both the KV-MM and the ACD-MM
algorithms provide the same maximum number of decoding outputs, it is required

$$l_{\text{KV-MM}} = 2^n l_{\text{ACD-MM}}. \quad (63)$$

Recalling Corollaries 9 - 10 and 14 - 15, the KV-MM complexity will be $2^{4n}$ times of the ACD-
MM complexity. On the other hand, if the two ASD algorithms have the same complexity, it

\[\text{Note that the complexity is drawn under } m \leq l \text{ and } m \text{ is replaced by } l. \text{ This characterization is more accurate when } m \text{ and } l \text{ are sufficiently large.} \]
is required that

\[ l_{KV-MM} = 2^n_l_{ACD-MM}. \]  

(64)

For example, with \( \eta = 5 \), the KV-MM algorithm will have a similar complexity as the ACD-MM algorithm if \( l_{KV-MM} = 2l_{ACD-MM} \).

VI. SIMULATION RESULTS AND DISCUSSION

This section presents simulation results of the KV-MM and the ACD-MM algorithms. They are obtained over the additive white Gaussian noise (AWGN) channel using binary phase shift keying (BPSK) modulation. The two algorithms are compared under the benchmarks of the maximum decoding output cardinality and the decoding complexity. The numerical complexity results are measured as the average number of finite field arithmetic operations that is required to decode a codeword, including the factorization and the re-encoding transform when it is applied. Besides comparing the two ASD algorithms, we also show their complexity difference in using the MM interpolation and Koetter’s interpolation. We denote the KV and the ACD algorithms that employ Koetter’s interpolation as the KV-Koetter and the ACD-Koetter algorithms, respectively.

Fig. 2 compares the KV-MM and ACD-MM performance in decoding the (63, 31) RS code.

Fig. 2 compares the KV-MM and ACD-MM performance in decoding the (63, 31) RS code. With \( m = 1 \), the ACD-MM of \( \eta = 2, 3, 4 \) yields the same decoding output cardinality as the KV-MM of \( l = 4, 8, 16 \), respectively. Our results show that with the same decoding output cardinality,
the KV-MM algorithm outperforms the ACD-MM algorithm. However, based on our discussion in Section V.C, the KV-MM algorithm is more complex. Tables I and II show the KV-MM and ACD-MM complexity in decoding the RS code, respectively. It can be seen that with the same decoding output cardinality, the KV-MM algorithm is more complex than the ACD-MM algorithm despite whether the re-encoding transform is applied. For example, the KV-MM algorithm with $l = 4$ is more complex than the ACD-MM algorithm with $(m, \eta) = (1, 2)$. For the KV algorithm, using MM interpolation yields a much lower complexity than the case using Koetter’s interpolation. A similar result can be observed for the ACD algorithm. However, for the re-encoding transformed ACD algorithm with $m = 1$, using the MM interpolation is more complex than using Koetter’s interpolation. This is because Koetter’s interpolation allows Gröbner basis of all test-vectors to be generated in a binary tree growth fashion, eliminating the redundant computation between decoding trials. But this interpolation scheduling advantage disappears when $m$ increases, e.g., when $m = 5$, the ACD-MM algorithm is simpler than the ACD-Koetter algorithm. Moreover, Table II also shows that when $m = 1$, the re-encoding transformed ACD-MM algorithm is more complex than its prototype. This is introduced by the test-vector transform of (51) - (54). The MM complexity reduction brought by the transform cannot compensate this extra computation.
Again, when \( m \) increases, a larger module will be formed. The re-encoding transform can still show its complexity reduction effect. Moreover, in contrast to the ACD-Koetter algorithm [11], the ACD-MM algorithm can perform the interpolation of each test-vector in parallel, leveraging the decoding latency to that of a single decoding trial.

![Diagram](image)

**Fig. 3.** ACD-MM performance in decoding the (63, 31) RS code.

Based on (64), we know that the ACD-MM algorithm with \( m = 1, \eta = 10 \) yields a similar complexity as the KV-MM algorithm with \( l = 4 \). Fig. 2 shows that they also yield a similar decoding performance. As for the impact of re-encoding, Tables I and II show that a complexity reduction factor of \( \frac{k}{\eta} \) can be realized. Fig. 3 further shows the ACD-MM decoding performance for the (63, 31) RS code with different interpolation multiplicities. With \( m = 1 \) and \( m = 5 \), each Chase decoding trial can correct at most 16 and 18 symbol errors, respectively. Therefore, by increasing \( m \), the ACD-MM performance can be further enhanced. Again, Table II shows increasing \( m \) also increases the decoding complexity by order of magnitudes.

Fig. 4 shows the decoding performance of the popular (255, 239) RS code. Both the KV-MM and the ACD-MM algorithms apply the re-encoding transform. For the ACD-MM algorithm, \( m = 1 \). Their complexity results are shown in Table III. Fig. 4 again shows with the same decoding output cardinality, e.g., the KV-MM with \( l = 8 \) and the ACD-MM with \( \eta = 3 \), the KV-MM decoding performance prevails. On the other hand, with a similar decoding complexity, the
Fig. 4. KV-MM and ACD-MM performance comparison in decoding the (255, 239) RS code.

TABLE III

| $l$  | KV-MM          | KV-Koetter     | $\eta$ | ACD-MM         | ACD-Koetter     |
|------|----------------|----------------|--------|----------------|----------------|
| 4    | $1.63 \times 10^7$ | $1.06 \times 10^8$ | 2      | $7.77 \times 10^7$ | $2.28 \times 10^7$ |
| 8    | $1.27 \times 10^8$ | $3.24 \times 10^8$ | 3      | $8.35 \times 10^7$ | $2.43 \times 10^7$ |
| 16   | $9.51 \times 10^8$ | $2.51 \times 10^9$ | 4      | $9.51 \times 10^7$ | $2.71 \times 10^7$ |

ACD-MM algorithm with $\eta = 10$ significantly outperforms the KV-MM algorithm with $l = 4$. This performance gain is more remarkable than the one observed in decoding the half rate (63, 31) RS code. This is due to the fact that the KV decoding favors low rate codes. Table III further demonstrates that when the re-encoding transform is applied, MM interpolation can significantly reduce the complexity of the KV algorithm. This is not the case for the ACD algorithm with $m = 1$, in which Koetter’s interpolation can be less complex. Again, this thanks to its binary tree growth interpolation manner.

VII. Conclusions

This paper has introduced two ASD algorithms for RS codes utilizing the MM interpolation technique, the KV-MM and the ACD-MM algorithms. Unlike Koetter’s interpolation that gen-
erates the desirable Gröbner basis in a point-by-point fashion, the MM interpolation formulates a module and further reduces it into the weak Popov form, resulting in the desirable Gröbner basis. This interpolation technique is simpler and can be applied in various ASD algorithms of RS codes. The two algorithms’ re-encoding transformed variants have also been introduced. They have a simpler MM interpolation since the transform can reduce the size of module entries. Our analysis has shown that the MM complexity is $O(n(n-k)l^5)$ and $O((n-k)^2l^5)$ for the cases without and with the re-encoding transform, respectively. This conclusion implies that the MM interpolation complexity favors high rate codes and the complexity reduction brought by re-encoding is also more effective for high rate codes. These findings are of practical interest. Our simulation results have verified that the MM interpolation enables lighter decoding computation for the two ASD algorithms, despite whether the re-encoding transform is applied. On the other hand, a comprehensive decoding performance comparison between the KV and the ACD algorithms has been performed. With the same decoding output cardinality, the KV decoding performance prevails. However, with a similar decoding complexity, the ACD performance prevails for high rate codes. It is hoped that this article can provide more insight into the advanced RS decoding algorithms.

ACKNOWLEDGMENT

Part of this research was carried out during the second author’s sabbatical at University of Notre Dame, U.S., who is grateful to Prof. Daniel J. Costello’s support and discussions.

APPENDIX A

PROOF OF LEMMA 3

With the module generators defined by (31), $P_t(x, y\psi(x))$ can be written as

$$P_t(x, y\psi(x)) = \prod_{j=0}^{n-1} (x - \alpha_j)^{m_j(t)} \prod_{\varepsilon=0}^{t-1} (y\psi(x) - F_\varepsilon(x)).$$

(65)

Based on (36), we know

$$\prod_{j=0}^{n-1} (x - \alpha_j)^{m_j(t)} = \prod_{j \in \bar{\gamma}} (x - \alpha_j)^{m_j(t)} \prod_{j \in \gamma} (x - \alpha_j)^{m_j(t)}$$

$$= \phi(x) \cdot \prod_{j \in \bar{\gamma}} (x - \alpha_j)^{m_j(t)} \prod_{j \in \gamma} (x - \alpha_j)^{m_j(t) - m_j(0)}.$$  

(66)
Based on the $F_{\varepsilon}(x)$ of (35) and the fact that $w^{(e)}_j = 0, \forall j \in \Lambda_{\varepsilon}$, we have

$$y\psi(x) - F_{\varepsilon}(x)$$

$$= y\psi(x) - \sum_{j=0}^{n-1} w^{(e)}_j \prod_{j'=0, j' \neq j}^{n-1} \frac{x - \alpha_{j'}}{\alpha_j - \alpha_{j'}}$$

$$= y\psi(x) - \sum_{j \in T \cup \Lambda_{\varepsilon}} w^{(e)}_j \prod_{j' \in \Lambda_{\varepsilon}}^{n-1} \frac{x - \alpha_{j'}}{\alpha_j - \alpha_{j'}}.$$ 

Let us denote

$$\bar{w}^{(e)}_j = \frac{w^{(e)}_j}{\prod_{j'=0, j' \neq j}^{n-1} (\alpha_j - \alpha_{j'})},$$

then

$$y\psi(x) - F_{\varepsilon}(x)$$

$$= y\psi(x) - \sum_{j \in T \cup \Lambda_{\varepsilon}} \bar{w}^{(e)}_j \prod_{j' \in \Lambda_{\varepsilon}}^{n-1} (x - \alpha_{j'}).$$

$$= y\psi(x) - \prod_{j' \in \Lambda_{\varepsilon}} (x - \alpha_{j'}) \sum_{j \in T \cup \Lambda_{\varepsilon}} \bar{w}^{(e)}_j \prod_{j' \in \Lambda_{\varepsilon}}^{n-1} (x - \alpha_{j'}).$$

$$= y\psi(x) - \prod_{j' \in \Lambda_{\varepsilon}} (x - \alpha_{j'}) \cdot T_{\varepsilon}(x),$$

where

$$T_{\varepsilon}(x) = \sum_{j \in T \cup \Lambda_{\varepsilon}} \bar{w}^{(e)}_j \prod_{j' \in \Lambda_{\varepsilon}}^{n-1} (x - \alpha_{j'}).$$

Based on (37), we know $\psi(x) = \prod_{j \in \Lambda_{\varepsilon}} (x - \alpha_j) \prod_{j \in \Lambda_{\varepsilon}'} (x - \alpha_j),$  

$$y\psi(x) - \prod_{j \in \Lambda_{\varepsilon}} (x - \alpha_j) \cdot T_{\varepsilon}(x)$$

$$= \prod_{j \in \Lambda_{\varepsilon}} (x - \alpha_j) \cdot \left( y \prod_{j \in \Lambda_{\varepsilon}} (x - \alpha_j) - T_{\varepsilon}(x) \right).$$

Therefore,

$$\prod_{\varepsilon=0}^{t-1} (y\psi(x) - F_{\varepsilon}(x)) = \prod_{\varepsilon=0}^{t-1} \prod_{j \in \Lambda_{\varepsilon}} (x - \alpha_j) \cdot \prod_{\varepsilon=0}^{t-1} \left( y \prod_{j \in \Lambda_{\varepsilon}} (x - \alpha_j) - T_{\varepsilon}(x) \right).$$

(67)

We now derive an equivalent expression for $\prod_{\varepsilon=0}^{t-1} \prod_{j \in \Lambda_{\varepsilon}} (x - \alpha_j).$ Let us first define the transformed balanced list $L_j$ as

$$L_j = \{(\alpha_j, \bar{w}_j^{(0)}), (\alpha_j, \bar{w}_j^{(1)}), \ldots, (\alpha_j, \bar{w}_j^{(m_j-1)})\},$$
where $\bar{w}_j^{(e)} = \frac{y_j^{(e)} - \bar{H}(\alpha_j)}{\prod_{j' = 0, j' \neq j}^{m_j - 1} (\alpha_j - \alpha_{j'})}$. Partition it into two disjoint sets as

$$\mathcal{L}_j(t) = \{ (\alpha_j, \bar{w}_j^{(t)}), (\alpha_j, \bar{w}_j^{(1)}), \ldots, (\alpha_j, \bar{w}_j^{(t-1)}) \}$$

and

$$\mathcal{L}_j(t) = \{ (\alpha_j, \bar{w}_j^{(t)}) , (\alpha_j, \bar{w}_j^{(t+1)}), \ldots, (\alpha_j, \bar{w}_j^{(m_j-1)}) \}.$$ 

Let $\theta_j(t)$ denote the multiplicity of $(\alpha_j, \bar{w}_j^{(0)})$ in $\mathcal{L}_j(t)$. Since $\bar{w}_j^{(0)} = 0, \forall j \in \Upsilon$, $\bar{w}_j^{(0)} = 0$ and $\theta_j(t)$ is the multiplicity of $\bar{w}_j^{(0)}$ of $\mathcal{L}_j(t)$ and $j \in \Upsilon$. Since $\Lambda_\varepsilon = \{ j \mid w_j^{(e)} = 0, j \in \Upsilon \}$, we have

$$\prod_{\varepsilon = 0, j \in \Lambda_\varepsilon}^{t-1} \prod_{j \in \Upsilon} (x - \alpha_j) = \prod_{j \in \Upsilon} (x - \alpha_j)^{\theta_j(t)}. \tag{68}$$

Based on (66), (67) and (68), we have

$$P_t(x, y \psi(x)) = \phi(x) \cdot \prod_{j \in \Upsilon} (x - \alpha_j)^{m_j(t)} \cdot U_t(x) \cdot \prod_{\varepsilon = 0}^{t-1} \left( y \prod_{j \in \Lambda_\varepsilon} (x - \alpha_j) - T_\varepsilon(x) \right),$$

where

$$U_t(x) = \prod_{j \in \Upsilon} (x - \alpha_j)^{m_j(t) - m_j(0) + \theta_j(t)}.$$ 

$U_t(x)$ can be simplified as follows. Let $\chi_j(t)$ denote the multiplicity of $(\alpha_j, 0)$ in $\mathcal{L}_j(t)$, we have

$$m_j(0) = \theta_j(t) + \chi_j(t), \forall j \in \Upsilon.$$ 

For $j \in \Upsilon$, if $\bar{w}_j^{(t)} = \bar{w}_j^{(0)} = 0$, $\chi_j(t) = m_j(t)$ and $m_j(0) = \theta_j(t) + m_j(t)$. Otherwise, if $\bar{w}_j^{(t)} \neq 0$, $\chi_j(t) = m_j(t) - 1$ and $m_j(0) = \theta_j(t) + m_j(t) - 1$. Hence, when $j \in \Lambda_t, m_j(t) - m_j(0) + \theta_j(t) = 1$. When $j \in \Lambda_\varepsilon, m_j(t) - m_j(0) + \theta_j(t) = 0$. Therefore,

$$U_t(x) = \prod_{j \in \Lambda_t} (x - \alpha_j).$$

As a result,

$$P_t(x, y \psi(x)) = \phi(x) \cdot \prod_{j \in \Upsilon} (x - \alpha_j)^{m_j(t)} \cdot \prod_{j \in \Lambda_t} (x - \alpha_j) \cdot \prod_{\varepsilon = 0}^{t-1} \left( y \prod_{j \in \Lambda_\varepsilon} (x - \alpha_j) - T_\varepsilon(x) \right).$$

■
APPENDIX B

PROOF OF LEMMA 7

For the KV-MM algorithm, its generator of (31) can be written as

\[ P_t(x, y) = G_t(x) \cdot B_t(x, y), \]

where \( G_t(x) = \prod_{j=0}^{n-1} (x - \alpha_j)^{m_{j(t)}} \) and \( B_t(x, y) = \prod_{\varepsilon=0}^{t-1} (y - F_{\varepsilon}(x)) = \sum_{\tau=0}^{t} b_{\tau}^{(t)}(x)y^{\tau}. \) Further based on mapping (20), we have \( A_{\ell|t}^{(\tau)} = G_t(x) \cdot b_{\tau}^{(t)}(x) \cdot x^{(k-1)\tau}. \) Let \( \tau_t = \operatorname{arg\,max}\{ \deg A_{\ell|t}^{(\tau)}, \forall \tau \} \) identify the maximum entry of \( A_{\ell|t} \) such that \( \deg A_{\ell|t}^{(\tau)} \geq \deg A_{\ell|t}^{(\tau)}, \forall \tau \neq \tau_t. \) Therefore, \( \deg A_{\ell|t} = \deg(G_t(x) \cdot b_{\tau_t}^{(t)}(x) \cdot x^{(k-1)\tau_t}) \) and

\[ \deg A_t = \sum_{t=0}^{l} \deg A_{\ell|t} \]

\[ = \sum_{t=0}^{l} (\deg G_t(x) + \deg b_{\tau_t}^{(t)}(x) + (k-1)\tau_t). \]

Since \( A_t \) is a lower-triangle matrix and \( b_{\tau_t}^{(t)}(x) = 1, \) we have

\[ \deg \det A_t = \deg \left( \prod_{t=0}^{l} G_t(x) \cdot x^{(k-1)t} \right) \]

\[ = \sum_{t=0}^{l} (\deg G_t(x) + (k-1)t). \]

Therefore, \( \deg A_t - \deg \det A_t = \sum_{t=0}^{l} (\deg b_{\tau_t}^{(t)}(x) + (k-1)\tau_t - (k-1)t). \) Since \( \deg b_{\tau_t}^{(t)}(x) \leq (n-1)(t-\tau_t), \) we have

\[ \deg A_t - \deg \det A_t \leq \sum_{t=0}^{l} ((n-1)(t-\tau_t) + (k-1)\tau_t - (k-1)t). \]

Therefore, when \( \tau_t = 0, \) \( \max\{ \deg A_t - \deg \det A_t \} = \frac{1}{2}(n-k)(l^2 + l). \)

For the ACD-MM algorithm, when \( 0 \leq t \leq m, \) \( \deg A_t = nm - t. \) When \( m < t \leq l, \) \( \deg A_t = (n-1)t. \) Hence, we have

\[ \deg A_t = \sum_{t=0}^{m} (nm - t) + \sum_{t=m+1}^{l} (n-1)t \]

and

\[ \deg \det A_t = \deg \prod_{t=0}^{m} G^{m-t}(x) \cdot \prod_{t=0}^{l} x^{(k-1)t} \]

\[ = \sum_{t=0}^{m} n(m - t) + \sum_{t=0}^{l} (k-1)t. \]

Therefore, \( \deg A_t - \deg \det A_t = \sum_{t=0}^{l} (n-k)t = \frac{1}{2}(n-k)(l^2 + l). \) \( \blacksquare \)
REFERENCES

[1] E. Berlekamp, *Algebraic Coding Theory*. New York: McGraw-Hill, 1968.

[2] J. Massey, “Shift register synthesis and BCH decoding,” *IEEE Trans. Inform. Theory*, vol. 15, no. 1, pp. 122-127, Jan. 1969.

[3] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, “A method for solving key equation for decoding Goppa codes,” *Inform. Control*, vol. 27, no. 1, pp. 87-99, Feb. 1975.

[4] L. Welch and E. Berlekamp, “Error correction for algebraic block codes,” U.S. Patent, no. 4 633 470, Dec. 30, 1986.

[5] R. Koetter, “Fast generalized minimum-distance decoding of algebraic-geometric and Reed-Solomon codes,” *IEEE Trans. Inform. Theory*, vol. 42, no. 3, pp. 721-737, May 1996.

[6] E. Berlekamp, “Bounded distance +1 soft-decision Reed-Solomon decoding,” *IEEE Trans. Inform. Theory*, vol. 42, no. 3, pp. 704-720, May 1996.

[7] M. Sudan, “Decoding of Reed-Solomon codes beyond the error-correction bound,” *J. Complexity*, vol. 13, no. 1, pp. 180-193, Mar. 1997.

[8] V. Guruswami and M. Sudan, “Improved decoding of Reed-Solomon and algebraic-geometric codes,” *IEEE Trans. Inform. Theory*, vol. 45, no. 6, pp. 1757-1767, Sept. 1999.

[9] R. Koetter, “On algebraic decoding of algebraic-geometric and cyclic codes,” Ph.D dissertation, Univ. Linköping, Linköping, Sweden, 1996.

[10] R. Koetter and A. Vardy, “Algebraic soft-decision decoding of Reed-Solomon codes,” *IEEE Trans. Inform. Theory*, vol. 49, no. 11, pp. 2809-2825, Nov. 2003.

[11] J. Bellorado and A. Kavčić, “Low-complexity soft-decoding algorithms for Reed-Solomon codes—Part I: An algebraic soft-in hard-out Chase decoder,” *IEEE Trans. Inform. Theory*, vol. 56, no. 3, pp. 945-959, Mar. 2010.

[12] G. Schmidt, V. Sidorenko, and M. Bossert, “Syndrome decoding of Reed-Solomon codes beyond half the minimum distance based on shift-register synthesis,” *IEEE Trans. Inform. Theory*, vol. 56, no. 10, pp. 5245-5252, Oct. 2010.

[13] M. Fossorier and S. Lin, “Soft-decision decoding of linear block codes based on ordered statistics,” *IEEE Trans. Inform. Theory*, vol. 41, no. 5, pp. 1379-1396, Sept. 1995.

[14] A. Vardy and Y. Be’ery, “Bit-level soft-decision decoding of Reed-Solomon codes,” *IEEE Trans. Commun.*, vol. 39, no. 3, pp. 440-444, Mar. 1991.

[15] V. Ponnampalam and B. Vucetic, “Soft decision decoding of Reed-Solomon codes,” *IEEE Trans. Commun.*, vol. 50, no. 11, pp. 1758-1768, Nov. 2002.

[16] J. Jiang and K. Narayanan, “Iterative soft-input soft-output decoding of Reed-Solomon codes by adapting the parity-check matrix,” *IEEE Trans. Inform. Theory*, vol. 52, no.8, pp. 3746-3756, Aug. 2006.

[17] M. El-Khamy and R. McEliece, “Iterative algebraic soft-decision list decoding of Reed-Solomon codes,” *IEEE J. Sel. Areas Commun.*, vol. 24, no. 3, pp. 481-490, Mar. 2006.

[18] R. Koetter, J. Ma, and A. Vardy, “The re-encoding transformation in algebraic list-decoding of Reed-Solomon codes,” *IEEE Trans. Inform. Theory*, vol. 57, no. 2, pp. 633-647, Feb. 2011.

[19] Y. Wu, “New list decoding algorithms for Reed-Solomon and BCH codes,” *IEEE Trans. Inform. Theory*, vol. 54, no. 8, pp. 3611-3630, Aug. 2008.

[20] P. Trifonov and M. Lee, “Efficient interpolation in the Wu list decoding algorithm,” *IEEE Trans. Inform. Theory*, vol. 58, no. 9, pp. 5963-5971, Sept. 2012.

[21] Y. Cassuto, J. Bruck, and R. McEliece, “On the average complexity of Reed-Solomon list decoders,” *IEEE Trans. Inform. Theory*, vol. 59, no. 4, pp. 2336-2351, Apr. 2013.
[22] L. Chen, S. Tang, and X. Ma, “Progressive algebraic soft-decision decoding of Reed-Solomon codes,” *IEEE Trans. Commun.*, vol. 61, no. 2, pp. 433-442, Feb. 2013.

[23] K. Lee and M. O’Sullivan, “List decoding of Reed-Solomon codes from a Gröbner basis perspective,” *J. Symb. Comput.*, vol. 43, no. 9, pp. 645-658, Sept. 2008.

[24] P. Beelen and K. Brander, “Key equations for list decoding of Reed-Solomon codes and how to solve them,” *J. Symb. Comput.*, vol. 45, no. 7, pp. 773-786, Jul. 2010.

[25] T. Mulders and A. Storjohann, “On lattice reduction for polynomial matrices,” *J. Symb. Comput.*, vol. 35, no. 4, pp. 377-401, Apr. 2003.

[26] M. Alekhnovich, “Linear diophantine equations over polynomials and soft decoding of Reed-Solomon codes,” *IEEE Trans. Inform. Theory*, vol. 51, no. 7, pp. 2257-2265, Jul. 2005.

[27] J. Nielsen, “List decoding of algebraic codes,” PhD dissertation, Dept. Applied Math. and Comput. Sci. Technical Univ. Denmark, 2013.

[28] P. Beelen, T. Hoholdt, J. Nielsen, and Y. Wu, “On rational interpolation-based list-decoding and list-decoding binary Goppa codes,” *IEEE Trans. Inform. Theory*, vol. 59, no. 6, pp. 3269-3281, June 2013.

[29] J. Nielsen, “Power decoding Reed–Solomon codes up to the Johnson radius,” submitted to *Advances in Mathematics of Communications*, arXiv preprint:1505.02111, 2016.

[30] J. Nielsen and A. Zeh, “Multi-trial Guruswami-Sudan decoding for generalised Reed-Solomon codes,” *Des. Codes Cryptogr.*, vol. 73, no. 2, pp. 507-527, Nov. 2014.

[31] K. Lee and M. O’Sullivan, “An interpolation algorithm using Gröbner bases for soft-decision decoding of Reed-Solomon codes,” in *Proc. the IEEE Int. Symp. Inform. Theory (ISIT)*, pp. 2032-2036, Seattle, U.S.A., Jul. 2006.

[32] J. Ma and A. Vardy, “A complexity reducing transformation for the Lee-O’Sullivan interpolation algorithm,” in *Proc. the IEEE Int. Symp. Inform. Theory (ISIT)*, pp. 1986-1990, Nice, France, Jun. 2007.

[33] L. Chen and M. Bossert, “Algebraic Chase decoding of Reed-Solomon codes using module minimisation,” in *Proc. the Int. Symp. Inform. Theory and Appl. (ISITA)*, pp. 305-309, Monterey, U.S.A. Oct. 2016.

[34] R. Roth and G. Ruckenstein, “Efficient decoding of Reed-Solomon codes beyond half the minimum distance,” *IEEE Trans. Inform. Theory*, vol. 46, no. 1, pp. 246-257, Jan. 2000.