IET Communications

Research Article

Module minimisation based low-complexity soft decoding of Reed–Solomon codes

ISSN 1751-8628
Received on 18th January 2019
Revised 27th August 2019
Accepted on 16th September 2019
doi: 10.1049/iet-com.2019.0064
www.ietdl.org

Jiongyue Xing[1], *Li Chen*[1] ✉, *Martin Bossert*[2]
[1]*School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou 510006, People's Republic of China*
[2]*Institute of Communications Engineering, Ulm University, Ulm 89073, Germany*
✉ *E-mail: chenli55@mail.sysu.edu.cn*

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, the algebraic soft decoding (ASD) further improves the decoding performance. This paper presents a unified study of two classical ASD algorithms, the algebraic Chase decoding and the Koetter-Vardy decoding. Their computationally expensive interpolation is solved by the module minimisation (MM) technique which consists of basis construction and basis reduction. Compared with Koetter's interpolation, the MM interpolation yields a smaller computational cost for the two ASD algorithms. Re-encoding transform is further applied to reduce the decoding complexity by reducing the degree of module generators. Based on assessing the degree of module seeds, a complexity reducing approach is introduced to further facilitate the two ASD algorithms. Computational cost of the two algorithms as well as their re-encoding transformed variants will be analysed. Performance of the two ASD algorithms will be compared under decoding expenditure benchmark, providing more practical insights of their applications.

1 Introduction

Reed–Solomon (RS) codes are widely employed in wireless communications and storage systems, in which the Berlekamp–Massey (BM) decoding algorithm [1, 2] is used. It is a syndrome-based decoding that delivers at most one decoded message. It has an efficient running time but their error-correction capability is limited by half of the code's minimum Hamming distance. Assisted by soft information to perform the error-erasure decoding, the generalised minimum-distance (GMD) decoding algorithm improves the decoding performance [3].

In late 90 s, Sudan introduced an interpolation-based algebraic decoding algorithm to correct errors beyond the above limit [4]. But this improvement only applies to low rate codes. Guruswami and Sudan later improved it to decode all rate codes up to the Johnson bound [5]. This 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 that constructs a minimum polynomial $Q(x, y)$ and root-finding that retrieves y-roots of $Q(x, y)$. Interpolation is often realised by Koetter's iterative polynomial construction algorithm [6, 7], which dominates the decoding complexity. It yields a Gröbner basis from which the minimum candidate is chosen as $Q(x, y)$. Koetter and Vardy later introduced the algebraic soft decoding (ASD), namely the KV algorithm [8]. It transforms soft received information into multiplicity information that defines the interpolation, outperforming the GS algorithm. The other classical ASD algorithm is the algebraic Chase decoding (ACD) [9]. It constructs a number of decoding test-vectors, whose formulation allows the following Koetter's interpolation to be performed in a binary tree growth fashion, resulting in a low decoding complexity. By further arranging the test-vectors such that the adjacent test-vectors only differ one symbol, the backward–forward (BF) interpolation can be applied to obtain a hardware friendly BF-ACD decoder [10]. Meanwhile, Wu proposed the algebraic list decoding that utilises the BM decoding output to construct $Q(x, y)$, leading to a lower decoding complexity [11]. Other complexity reducing approaches include the re-encoding transform [12, 13] and the progressive interpolation [14].

The interpolation problem can also be solved from the perspective of Gröbner basis of module [15, 16]. A module basis is first constructed, which contains bivariate polynomials that interpolate all prescribed points with their multiplicity. Presenting it as a matrix over univariate polynomials, row operation further reduces it into the Gröbner basis that is defined under a weighted monomial order. The minimum candidate of the basis is $Q(x, y)$. This interpolation technique is called module minimisation (MM) which consists of basis construction and basis reduction. Lee and O'Sullivan gave an explicit module basis construction for the GS and the KV algorithms in [16, 17], respectively. Ma and Vardy further defined the explicit module generators for the KV algorithm that applies the re-encoding transform [18]. For practical codes, the basis reduction can be efficiently realised by the Mulders–Storjohann (MS) algorithm [19]. Besides, there also exist several asymptotically faster basis reduction approaches [20–22], among which the Alekhnovich algorithm [20] can be seen as the divide-and-conquer variant of the MS algorithm. The MM interpolation has also been generalised to perform Wu's list decoding [23], the multi-trial GS decoding [24], the ACD [25] and the power decoding [26]. It has been shown that the MM interpolation yields a significantly lower complexity for the ASD algorithms than using Koetter's interpolation [18, 25].

However, despite its complexity advantage over Koetter's interpolation, these MM interpolation-based ASD algorithms have not been fully researched. Many practical aspects of this approach demand a more comprehensive understanding. On one hand, the exact complexity reduction yielded by the MM technique and the re-encoding transform is still unknown. On the other hand, the ACD and KV performances have not yet been compared under the decoding expenditure benchmark. Aiming to facilitate the application of the two MM interpolation-based soft decoding algorithms, this paper presents a comprehensive and unified study of them. From this point onwards, they are named the ACD-MM algorithm and the KV-MM algorithm, respectively. This work contributes in the following aspects:

- The ACD-MM algorithm and its re-encoding transformed variant are first introduced. In comparison to the ACD algorithm that employs Koetter's interpolation [9], the ACD-MM algorithm not only yields a lower complexity but also leverages the decoding latency to a single decoding event.
- For the KV-MM algorithm and its re-encoding transformed variant, their module basis construction is underpinned by the point

enumeration. A simple proof for their module generators will also be given.

• A complexity reducing approach will be further introduced for both of the ASD algorithms. This is realised by assessing the degree of module seeds. It leads to a significant complexity reduction, especially in the high signal-to-noise ratio (SNR) region.

• Complexity of the ACD-MM and the KV-MM algorithms will be analysed, showing the MM interpolation and the re-encoding transform are more effective in yielding a low complexity for high rate codes. This finding falls into the interest of practice.

• Finally, simulation results on complexity and decoding performances will be provided, showing the MM interpolation leads to a lower decoding complexity. The proposed complexity reducing approach can further result in a lower decoding cost. More importantly, performance of the two ASD algorithms will be compared under the decoding expenditure benchmark, providing more practical insights.

The rest of this paper is organised as follows. Section 2 introduces RS codes and the MM based GS decoding. Section 3 introduces the ACD-MM algorithm and its re-encoding transformed variant. Section 4 introduces the KV-MM algorithm and its re-encoding transformed variant. Section 5 introduces a complexity reducing approach for the ASD algorithms. Section 6 analyses the decoding complexity and Section 7 provides the simulation results. Finally, Section 8 concludes the paper.

2 RS codes and the MM-based GS decoding

This section introduces the prerequisites of the paper, including the RS codes and the MM-based GS decoding.

2.1 RS codes

Let $\mathbb{F}_q = \{\sigma_0, \sigma_1, \dots, \sigma_{q-1}\}$ 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, message polynomial $f(x) \in \mathbb{F}_q[x]$ is

$$f(x) = f_0 + f_1 x + \cdots + f_{k-1} x^{k-1}, \tag{1}$$

where $f_0, f_1, \dots, f_{k-1}$ are message symbols. Codeword $\underline{c} = (c_0, c_1, \dots, c_{n-1}) \in \mathbb{F}_q^n$ is generated by

$$\underline{c} = (f(\alpha_0), f(\alpha_1), \dots, f(\alpha_{n-1})), \tag{2}$$

where $\alpha_0, \alpha_1, \dots, \alpha_{n-1}$ are the n distinct non-zero elements of $\mathbb{F}_q$. They are called the code locators.

2.2 MM-based GS decoding

Let $\underline{\omega} = (\omega_0, \omega_1, \dots, \omega_{n-1}) \in \mathbb{F}_q^n$ denote the received word. The Hamming distance between $\underline{c}$ and $\underline{\omega}$ is $d_\mathrm{H}(\underline{c}, \underline{\omega}) = |\{j \mid c_j \neq \omega_j, \forall j\}|$. The GS decoding algorithm consists of two steps, interpolation and root-finding. Interpolation constructs the minimum polynomial $Q(x, y)$ that interpolates the n points $(\alpha_0, \omega_0), (\alpha_1, \omega_1), \dots, (\alpha_{n-1}, \omega_{n-1})$ with a prescribed multiplicity. Given $Q(x, y) = \sum_{a,b} Q_{ab} x^a y^b \in \mathbb{F}_q[x, y]$, its monomials $x^a y^b$ can be organised under the (μ, ν)-revlex order. The (μ, ν)-weighted degree of $x^a y^b$ is $\deg_{\mu,\nu} x^a y^b = \mu a + \nu b$. Given $x^{a_1} y^{b_1}$ and $x^{a_2} y^{b_2}$, it is claimed $x^{a_1} y^{b_1} < x^{a_2} y^{b_2}$, if $\deg_{\mu,\nu} x^{a_1} y^{b_1} < \deg_{\mu,\nu} x^{a_2} y^{b_2}$, or $\deg_{\mu,\nu} x^{a_1} y^{b_1} = \deg_{\mu,\nu} x^{a_2} y^{b_2}$ and $b_1 < b_2$. Let $x^{a'} y^{b'}$ denote the leading monomial (LM) of Q as $\mathrm{LM}(Q) = x^{a'} y^{b'}$, the (μ, ν)-weighted degree of Q is $\deg_{\mu,\nu} Q = \deg_{\mu,\nu} x^{a'} y^{b'}$. Furthermore, given polynomials Q_1 and Q_2 with leading monomials $x^{a_1'} y^{b_1'}$ and $x^{a_2'} y^{b_2'}$, respectively, $Q_1 < Q_2$ if $x^{a_1'} y^{b_1'} < x^{a_2'} y^{b_2'}$. The following GS decoding theorem can be introduced [5].

Theorem 1: For an (n,k) RS code, let $Q \in \mathbb{F}_q[x, y]$ denote a polynomial that interpolates the n points with a multiplicity of m. If $m(n - d_\mathrm{H}(\underline{c}, \underline{\omega})) > \deg_{1,k-1} Q(x, y)$, $Q(x, f(x)) = 0$.

Therefore, interpolation constructs Q with the minimum $(1, k-1)$-weighted degree, and $f(x)$ can be recovered by finding its y-roots [27]. Hence, the maximum decoding output list size (OLS) is $\deg_y Q$. In this paper, let $l = \deg_y Q$ denote the decoding parameter. Note that $m \leq l$ in the GS algorithm [5].

Definition 1: Let $\underline{\xi} = (\xi_0(x), \xi_1(x), \dots, \xi_l(x))$ denote a vector over $\mathbb{F}_q[x]$, the degree of $\underline{\xi}$ is

$$\deg \underline{\xi} = \max\{\deg \xi_\tau(x), \forall \tau\}. \tag{3}$$

The leading position (LP) of $\underline{\xi}$ is

$$\mathrm{LP}(\underline{\xi}) = \max\{\tau \mid \deg \xi_\tau(x) = \deg \underline{\xi}\}. \tag{4}$$

Definition 2: Given a matrix $\mathscr{V}$ over $\mathbb{F}_q[x]$, its row-t and entry of row-t column-τ are denoted by $\mathscr{V}|_t$ and $\mathscr{V}|_t^{(\tau)}$, respectively. Furthermore, the degree of $\mathscr{V}$ is

$$\deg \mathscr{V} = \sum_t \deg \mathscr{V}\Big|_t. \tag{5}$$

Definition 3: Module $\mathscr{M}_l$ is the space of all polynomials over $\mathbb{F}_q[x, y]$ that interpolate all prescribed points with their multiplicity and have a maximum y-degree of l.

The GS decoding using MM interpolation can now be described, which consists of basis construction and basis reduction. Let us define two *module seeds*

$$G(x) = \prod_{j=0}^{n-1} (x - \alpha_j) \tag{6}$$

and

$$R(x) = \sum_{j=0}^{n-1} \omega_j \Phi_j(x), \tag{7}$$

where

$$\Phi_j(x) = \prod_{j'=0, j' \neq j}^{n-1} \frac{x - \alpha_{j'}}{\alpha_j - \alpha_{j'}} \tag{8}$$

is the Lagrange basis polynomial. It satisfies $\Phi_j(\alpha_j) = 1$ and $\Phi_j(\alpha_{j'}) = 0, \forall j' \neq j$. As a result, $R(\alpha_j) = \omega_j, \forall j$. With a multiplicity of m and a decoding OLS of l, $\mathscr{M}_l$ can be generated as an $\mathbb{F}_q[x]$-module by the following $l + 1$ polynomials [28]

$$P_t(x, y) = G(x)^{m-t} (y - R(x))^t, \text{if } 0 \leq t \leq m, \tag{9}$$

$$P_t(x, y) = y^{t-m} (y - R(x))^m, \text{if } m < t \leq l. \tag{10}$$

Note that $P_t(\alpha_j, \omega_j) = 0, \forall (t, j)$ and the total exponents of $G(x)$ and $y - R(x)$ is m. Since any element of $\mathscr{M}_l$ can be presented as an $\mathbb{F}_q[x]$-linear combination of $P_t(x, y)$, (9) and (10) construct a basis of module $\mathscr{M}_l$, denoted as $\mathscr{B}_l$ [24]. Moreover, since $P_t(x, y) = \sum_{\tau \leq t} P_t^{(\tau)}(x) y^\tau$ where $P_t^{(\tau)}(x) \in \mathbb{F}_q[x]$, $\mathscr{B}_l$ can be presented as an $(l+1) \times (l+1)$ matrix over $\mathbb{F}_q[x]$ by letting $\mathscr{B}_l|_t^{(\tau)} = P_t^{(\tau)}(x), \forall (t, \tau)$. Each row of the matrix corresponds to a bivariate polynomial of $\mathscr{B}_l$.

Definition 4: Assume that $\{g_t \in \mathbb{F}_q[x, y], 0 \leq t \leq l\}$ generates module $\mathscr{M}_l$. Under the (μ, ν)-revlex order, if y-degree of $\mathrm{LM}(g_t)$ is different, $\{g_t \in \mathbb{F}_q[x, y], 0 \leq t \leq l\}$ is a Gröbner basis of $\mathscr{M}_l$ [17].

The constructed basis $\mathscr{B}_l$ will then be reduced into the Gröbner basis. In this paper, the MS algorithm [19] is utilised to perform the basis reduction. In literature, there exist several asymptotically faster approaches [20–22]. However, they cannot show their efficiency for the practical codes. For example, the Alekhnovich algorithm [20] is only faster than the MS algorithm when codeword length is longer than 2 000 [24].

Definition 5: Given a square matrix $\mathscr{V}$ over $\mathbb{F}_q[x]$, if any of its two rows $\mathscr{V}|_t$ and $\mathscr{V}|_{t'}$ exhibit $\mathrm{LP}(\mathscr{V}|_t) \neq \mathrm{LP}(\mathscr{V}|_{t'})$, it is in the *weak Popov form* [19].

Let $\mathscr{D}_{\beta, l} = \mathrm{diag}(1, x^{\beta}, \ldots, x^{l\beta})$ denote the diagonal matrix of size $(l+1) \times (l+1)$ and β is an integer. In decoding an (n,k) RS code, performing the mapping of

$$\mathscr{A}_l = \mathscr{B}_l \cdot \mathscr{D}_{k-1, l} \tag{11}$$

enables $\deg \mathscr{A}_l|_t = \deg_{1, k-1} P_t(x, y)$. The MS algorithm [19] further performs the row operation to reduce $\mathscr{A}_l$ into the weak Popov form $\mathscr{A}_l'$. Afterwards, perform the demapping of

$$\mathscr{B}_l' = \mathscr{A}_l' \cdot \mathscr{D}_{-(k-1), l}. \tag{12}$$

Lemma 1: Under the $(1, k-1)$-revlex order, $\mathscr{B}_l'$ is the Gröbner basis of $\mathscr{M}_l$.

Proof: Based on $\mathscr{B}_l'|_t$, $P_t'(x, y) = \sum_{\tau \le l} P_t'^{(\tau)}(x) y^{\tau}$ can be constructed by letting $P_t'^{(\tau)}(x) = \mathscr{B}_l'|_t^{(\tau)}$. Since $\deg_{1, k-1} P_t'(x, y) = \deg \mathscr{A}_l'|_t = \deg \mathscr{A}_l'|_t^{(\mathrm{LP}(\mathscr{A}_l'|_t))} = \deg P_t'^{(\mathrm{LP}(\mathscr{A}_l'|_t))}(x) + (k-1) \cdot \mathrm{LP}(\mathscr{A}_l'\big|_t)$, $\mathrm{LM}(P_t'(x, y)) = P_t'^{(\mathrm{LP}(\mathscr{A}_l'|_t))}(x) y^{\mathrm{LP}(\mathscr{A}_l'|_t)}$. When $\mathscr{A}_l'$ is in the weak Popov form, y-degree of $\mathrm{LM}(P_t'(x, y))$, i.e. $\mathrm{LP}(\mathscr{A}_l'|_t)$, is different. Based on Definition 4, $\mathscr{B}_l'$ is the Gröbner basis. □

Let $\mathscr{A}_l'|_{t^*}$ denote the row that has the minimum degree, the interpolated polynomial $Q(x, y) = \sum_{\tau \le l} Q^{(\tau)}(x) y^{\tau}$ can be constructed from $\mathscr{B}_l'|_{t^*}$ by letting

$$Q^{(\tau)}(x) = \mathscr{B}_l'|_{t^*}^{(\tau)}, \forall \tau. \tag{13}$$

Finally, determine the y-roots of Q using the recursive coefficient search algorithm [27].

3 ACD-MM algorithm

This section introduces the ACD-MM algorithm. It first constructs a number of decoding test-vectors based on the reliability matrix. The GS decoding will be further performed on each test-vector. Its re-encoding transformed variant will also be introduced.

3.1 From reliability matrix to test-vectors

Assume codeword $\underline{c}$ is transmitted through a memoryless channel and $\underline{r} = (\mathrm{r}_0, \mathrm{r}_1, \ldots, \mathrm{r}_{n-1}) \in \mathbb{R}^n$ is the received symbol vector, where $\mathbb{R}$ denotes the channel output alphabet. The channel observation is represented by a reliability matrix $\mathbf{\Pi}$ whose entries are the *a posteriori* probability defined as $\pi_{ij} = \Pr[c_j = \sigma_i \mid \mathrm{r}_j]$, where $0 \le i \le q-1$ and $0 \le j \le n-1$. Note that it is assumed $\Pr[c_j = \sigma_i] = (1/q), \forall (i, j)$. Let $i_j^{\mathrm{I}} = \arg\max_i \{\pi_{ij}\}$ and $i_j^{\mathrm{II}} = \arg\max_{i, i \neq i_j^{\mathrm{I}}} \{\pi_{ij}\}$ denote the row indices of the largest and the second largest entries of column j, respectively. The two most likely decisions for c_j are $r_j^{\mathrm{I}} = \sigma_{i_j^{\mathrm{I}}}$ and $r_j^{\mathrm{II}} = \sigma_{i_j^{\mathrm{II}}}$. Define the symbol wise reliability metric as

$$\gamma_j = \frac{\pi_{i_j^{\mathrm{II}} j}}{\pi_{i_j^{\mathrm{I}} j}}, \tag{14}$$

where $\gamma_j \in (0, 1)$ [9]. With $\gamma_j \to 0$, the decision on c_j is more reliable, and vice versa. By sorting γ_j in an ascending order, a refreshed symbol index sequence $j_0, j_1, \ldots, j_{n-1}$ can be obtained, which indicates $\gamma_{j_0} < \gamma_{j_1} < \cdots < \gamma_{j_{n-1}}$. Choose η least reliable symbols that can be realised as either r_j^{I} or r_j^{II}, where $\eta < n$. For the remaining $n - \eta$ reliable symbols, they will be realised as r_j^{I}. Therefore, the interpolation test-vectors can be generally written as

$$\underline{\boldsymbol{r}}_u = (r_{j_0}^{(u)}, r_{j_1}^{(u)}, \ldots, r_{j_{k-1}}^{(u)}, r_{j_k}^{(u)}, \ldots, r_{j_{n-1}}^{(u)}), \tag{15}$$

where $u = 1, 2, \ldots, 2^{\eta}$, $r_j^{(u)} = r_j^{\mathrm{I}}$ for $j = j_0, j_1, \ldots, j_{n-\eta-1}$, and $r_j^{(u)} = r_j^{\mathrm{I}}$ or r_j^{II} for $j = j_{n-\eta}, j_{n-\eta+1}, \ldots, j_{n-1}$.

3.2 Basis construction and reduction

For each test-vector, the MM-based GS decoding that is described in Section 2.2 will be performed. In particular, given a test-vector $\underline{\boldsymbol{r}}_u$, polynomial $R(x)$ of (7) becomes

$$R_u(x) = \sum_{j=0}^{n-1} r_j^{(u)} \Phi_j(x). \tag{16}$$

Consequently, $R_u(\alpha_j) = r_j^{(u)}, \forall j$. Basis $\mathscr{B}_l$ can be constructed using the $l+1$ polynomials of (9) and (10), in which $R(x)$ is replaced by $R_u(x)$.

Note that 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 in a binary tree growth fashion [9].

The ACD-MM algorithm is summarised in Fig. 1, where $\hat{f}(x)$ is the decoding estimation of the message $f(x)$.

3.3 Re-encoding transformed ACD-MM algorithm

Re-encoding transforms the test-vectors so that they have at least k zero symbols. This will reduce the x-degree of module generators, resulting in a simpler MM interpolation. Let $\Theta = \{j_0, j_1, \ldots, j_{k-1}\}$ denote the index set of the k most reliable symbols. Its complementary set is $\bar{\Theta} = \{j_k, j_{k+1}, \ldots, j_{n-1}\}$. Let $\eta \le n - k$ so that the test-vectors would share at least k common symbols $r_{j_0}^{\mathrm{I}}, r_{j_1}^{\mathrm{I}}, \ldots, r_{j_{k-1}}^{\mathrm{I}}$. The k re-encoding points are $(\alpha_j, r_j^{\mathrm{I}})$ where $j \in \Theta$. The re-encoding polynomial is

$$H_{\Theta}(x) = \sum_{j \in \Theta} r_j^{\mathrm{I}} \prod_{j' \in \Theta, j' \neq j} \frac{x - \alpha_{j'}}{\alpha_j - \alpha_{j'}}. \tag{17}$$

Hence, $H_{\Theta}(\alpha_j) = r_j^{\mathrm{I}}, \forall j \in \Theta$. All test-vectors $\underline{\boldsymbol{r}}_u$ are transformed by

$$\underline{\boldsymbol{r}}_u \mapsto \underline{\boldsymbol{z}}_u : z_j^{(u)} = r_j^{(u)} - H_{\Theta}(\alpha_j), \forall j. \tag{18}$$

Consequently, the transformed test-vectors can be generally written as

```
Input: $\mathbf{\Pi}, \eta, m, l$;
Output: $\hat{f}(x)$;

1: Determine metrics $\gamma_j$ as in (14);
2: Formulate $2^{\eta}$ test-vectors $\underline{r}_u$ as in (15);
3: For each test-vector $\underline{r}_u$ do
4:     Formulate $\mathcal{B}_l$ by (9) (10) and map to $\mathcal{A}_l$ by (11);
5:     Reduce $\mathcal{A}_l$ into $\mathcal{A}_l'$ and demap it to $\mathcal{B}_l'$ by (12);
6:     Construct $Q$ by (13);
7:     Find $y$-roots of $Q$ to estimate $\hat{f}(x)$;
8: End for
```

Fig. 1 *Algorithm 1: the ACD-MM algorithm*

$$\underline{z}_u = (0, 0, \ldots, 0, z_{j_k}^{(u)}, \ldots, z_{j_{n-1}}^{(u)}) . \tag{19}$$

With a transformed test-vector $\underline{z}_u$, polynomial $R_u(x)$ of (16) is redefined as

$$R_u(x) = \sum_{j=0}^{n-1} z_j^{(u)} \Phi_j(x) . \tag{20}$$

Since $z_j^{(u)} = 0, \forall j \in \Theta$

$$V(x) = \prod_{j \in \Theta} (x - \alpha_j) \tag{21}$$

becomes the GCD for both $G(x)$ and the above $R_u(x)$. Therefore, given $\underline{z}_u$, two module seeds can be defined as

$$\tilde{G}(x) = \frac{G(x)}{V(x)} = \prod_{j \in \Theta} (x - \alpha_j) \tag{22}$$

and

$$\tilde{R}_u(x) = \frac{R_u(x)}{V(x)} = \sum_{j \in \Theta} \frac{z_j^{(u)}}{\varpi_j} \prod_{j' \in \Theta, j' \neq j} (x - \alpha_{j'}), \tag{23}$$

where $\varpi_j = \prod_{j'=0, j' \neq j}^{n-1} (\alpha_j - \alpha_{j'})$.

The following Lemma reveals the property of the module generators.

Lemma 2: Given a transformed test-vector $\underline{z}_u$ and a multiplicity m, $V(x)^m \mid P_t(x, yV(x))$ holds.

Proof: With the generators defined by (9) and (10), when $0 \leq t \leq m, P_t(x, yV(x))$ can be elaborated as

$$G(x)^{m-t}(-R_u(x))^t + \binom{t}{1} G(x)^{m-t}(-R_u(x))^{t-1} V(x) y$$
$$+ \cdots + G(x)^{m-t}(V(x)y)^t .$$

When $m < t \leq l, P_t(x, yV(x))$ becomes

$$(-R_u(x))^m (V(x)y)^{t-m} + \binom{m}{1} (-R_u(x))^{m-1} (V(x)y)^{t-m+1}$$
$$+ \cdots + (V(x)y)^t .$$

Since $V(x) \mid G(x)$ and $V(x) \mid R_u(x)$, it is straightforward to conclude that $V(x)^m \mid P_t(x, yV(x))$. □

Therefore, the following bijective mapping can be defined:

$$\begin{aligned} \varphi: \quad & \mathscr{M}_l \to \tilde{\mathscr{M}}_l \\ & P_t(x, y) \mapsto V(x)^{-m} P_t(x, yV(x)), \end{aligned} \tag{24}$$

where φ is an isomorphism between $\mathscr{M}_l$ and $\tilde{\mathscr{M}}_l$, i.e. $\tilde{\mathscr{M}}_l = \varphi(\mathscr{M}_l)$. Polynomials of $\tilde{\mathscr{M}}_l$ have lower x-degree than those of $\mathscr{M}_l$. Since the MS algorithm performs linear combination between its polynomials, the mapping of (24) will result in a simpler basis reduction process. Based on (9) and (10), the generators of $\tilde{\mathscr{M}}_l$ can be further obtained by

$$\tilde{P}_t(x, y) = \tilde{G}(x)^{m-t}(y - \tilde{R}_u(x))^t, \text{ if } 0 \leq t \leq m, \tag{25}$$

$$\tilde{P}_t(x, y) = (yV(x))^{t-m}(y - \tilde{R}_u(x))^m, \text{ if } m < t \leq l . \tag{26}$$

They form a basis $\tilde{\mathscr{B}}_l$ of $\tilde{\mathscr{M}}_l$. Again, $\tilde{\mathscr{B}}_l$ can be presented as a square matrix over $\mathbb{F}_q[x]$. Note that polynomials are now arranged under the $(1, -1)$-revlex order [12, 13]. However, performing

```
Input: $\mathbf{\Pi}, \eta, m, l$;
Output: $\hat{f}(x)$;

1: Determine metrics $\gamma_j$ as in (14) and define $\Theta$;
2: Formulate $2^\eta$ test-vectors $\underline{r}_u$ as in (15);
3: Transform all $\underline{r}_u$ into $\underline{z}_u$ as in (18);
4: For each test-vector $\underline{z}_u$ do
5:     Formulate $\tilde{\mathcal{B}}_l$ by (25) (26) and map to $\mathcal{A}_l$ by (27);
6:     Reduce $\mathcal{A}_l$ into $\mathcal{A}'_l$ and demap it to $\tilde{\mathcal{B}}'_l$ by (28);
7:     Construct $Q$ by (13) and (29);
8:     Find $y$-roots of $Q$ to estimate $\hat{f}(x)$;
9: End for
```

Fig. 2 *Algorithm 2: the re-encoding transformed ACD-MM algorithm*

$\mathscr{A}_l = \tilde{\mathscr{B}}_l \cdot \mathscr{D}_{-1,l}$ will cause some of the basis entries leaving $\mathbb{F}_q[x]$. Alternatively, let $\tilde{\mathscr{D}}_{\beta,l} = \text{diag}(x^{l\beta}, x^{(l-1)\beta}, \ldots, 1)$ and $\mathscr{A}_l$ will be generated by

$$\mathscr{A}_l = \tilde{\mathscr{B}}_l \cdot \tilde{\mathscr{D}}_{1,l}, \tag{27}$$

so that $\deg \mathscr{A}_l \big|_t = \deg_{1,-1} \tilde{P}_t(x, y) + l$. The MS algorithm will then reduce $\mathscr{A}_l$ into the weak Popov form $\mathscr{A}'_l$. Demap it by

$$\tilde{\mathscr{B}}'_l = \mathscr{A}'_l \cdot \tilde{\mathscr{D}}_{-1,l} \tag{28}$$

and polynomial $\tilde{Q}(x, y)$ can be retrieved from $\tilde{\mathscr{B}}'_l$ as in (13). Based on Lemma 2, the interpolated polynomial Q can be constructed by

$$Q(x, y) = V(x)^m \tilde{Q}\left(x, \frac{y}{V(x)}\right). \tag{29}$$

If $f'(x)$ is a y-root of Q, $\hat{f}(x)$ is estimated by $\hat{f}(x) = f'(x) + H_\Theta(x)$.

The re-encoding transformed ACD-MM algorithm is summarised in Fig. 2.

4 KV-MM algorithm

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

4.1 From reliability matrix to multiplicity matrix

The reliability matrix $\mathbf{\Pi}$ will be proportionally transformed into a multiplicity matrix $\boldsymbol{M}$ using Algorithm A of [8], where entry m_{ij} indicates the interpolation multiplicity for point (α_j, σ_i). Interpolation aims to find the minimum polynomial $Q(x, y)$ that interpolates all points (α_j, σ_i) with a multiplicity of m_{ij}. Let $i_j = \text{index}\{\sigma_i \mid \sigma_i = c_j\}$, the codeword score is defined as $S_M(\underline{c}) = \sum_{j=0}^{n-1} m_{i_j j}$. What follows is a sufficient condition for a successful KV decoding [8].

Theorem 2: For an (n,k) RS code, let $Q \in \mathbb{F}_q[x, y]$ denote an interpolated polynomial constructed based on $\boldsymbol{M}$. If $S_M(\underline{c}) > \deg_{1,k-1} Q(x, y)$, $Q(x, f(x)) = 0$.

The KV decoding is parameterised by the maximum decoding OLS of l and $\deg_y Q = l$. Given matrix $\boldsymbol{M}$, let us define

$$\mathsf{m}_j = \sum_{i=0}^{q-1} m_{ij} \tag{30}$$

and $\mathsf{m} = \max\{\mathsf{m}_j, \forall j\}$. The $\mathbf{\Pi} \to \boldsymbol{M}$ transform terminates when $\mathsf{m} = l$.

4.2 Basis construction and reduction

Based on Definition 3, the KV-MM algorithm generates a module $\mathscr{M}_l$ whose bivariate polynomials interpolate points (α_j, σ_i) with a multiplicity of at least m_{ij} and have a maximum y-degree of l. It can be underpinned by the following point enumeration.

Let L_j denote a list that enumerates interpolation points (α_j, σ_i) from column j of $\boldsymbol{M}$ as

$$L_j = [\underbrace{(\alpha_j, \sigma_i), \ldots, (\alpha_j, \sigma_i)}_{m_{ij}}, \forall i \text{ and } m_{ij} \neq 0]\,. \tag{31}$$

Note that $|L_j| = \mathsf{m}_j$. Its balanced list L_j' is further created by removing one of the most frequent elements in L_j to L_j'. Repeat this process m_j times until L_j is empty. Consequently, L_j' can be denoted as

$$L_j' = [(\alpha_j, y_j^{(0)}), (\alpha_j, y_j^{(1)}), \ldots, (\alpha_j, y_j^{(\mathsf{m}_j - 1)})], \tag{32}$$

where $y_j^{(0)}, y_j^{(1)}, \ldots, y_j^{(\mathsf{m}_j - 1)} \in \mathbb{F}_q$ and they may not be distinct. L_j' is a permutation of L_j and $|L_j'| = \mathsf{m}_j$. Let $\mathsf{m}_j(t)$ denote the maximum multiplicity of the last $\mathsf{m}_j - t$ elements of L_j' as

$$\mathsf{m}_j(t) = \max\{\text{multi.}((\alpha_j, y_j^{(\varepsilon)})) \mid \varepsilon = t, t+1, \ldots, \mathsf{m}_j - 1\}\,. \tag{33}$$

Note that $\mathsf{m}_j(0) = \max\{m_{ij}, \forall i\}$ and $\mathsf{m}_j(t) = 0$ for $t \geq \mathsf{m}_j$.

Now, it is sufficient to construct a basis of $\mathscr{M}_l$. First, the module seed is defined as

$$F_\varepsilon(x) = \sum_{j=0}^{n-1} y_j^{(\varepsilon)} \Phi_j(x), \tag{34}$$

where $\varepsilon = 0, 1, \ldots, l-1$. Based on (8), $F_\varepsilon(\alpha_j) = y_j^{(\varepsilon)}, \forall j$. Hence, $y - F_\varepsilon(x)$ interpolates points $(\alpha_j, y_j^{(\varepsilon)}), \forall j$. Now, $\mathscr{M}_l$ can be generated as an $\mathbb{F}_q[x]$-module by

$$P_t(x, y) = \prod_{j=0}^{n-1} (x - \alpha_j)^{\mathsf{m}_j(t)} \prod_{\varepsilon=0}^{t-1} (y - F_\varepsilon(x)), \tag{35}$$

where $t = 0, 1, \ldots, l$. It can be seen that $\prod_{\varepsilon=0}^{t-1}(y - F_\varepsilon(x))$ interpolates the first t points of all balanced lists, while $\prod_{j=0}^{n-1}(x - \alpha_j)^{\mathsf{m}_j(t)}$ interpolates the remaining points. Moreover, $\deg_y P_t(x, y) = t \leq l, \forall t$. Based on Definition 3, $P_t(x, y) \in \mathscr{M}_l$.

Lemma 3: Let $\mathscr{Q}_t(x, y) = \sum_{\tau=0}^{t} \mathscr{Q}_t^{(\tau)}(x) y^\tau \in \mathscr{M}_l$ with $\deg_y \mathscr{Q}_t = t$, $\prod_{j=0}^{n-1}(x - \alpha_j)^{\mathsf{m}_j(t)} \big| \mathscr{Q}_t^{(t)}(x)$ holds [20].

Consequently, the following Theorem can be proved.

Theorem 3: Any element of $\mathscr{M}_l$ can be written as an $\mathbb{F}_q[x]$-linear combination of $P_t(x, y)$.

Proof: Assume that $\mathscr{Q}(x, y) \in \mathscr{M}_l$ and let us write (35) as $P_t(x, y) = \sum_{\tau=0}^{t} P_t^{(\tau)}(x) y^\tau$. Since when $t = l$, $P_l^{(l)}(x) = 1$, there exists a polynomial $p_l(x) \in \mathbb{F}_q[x]$ that enables $\mathscr{Q}_{l-1}(x, y) = \mathscr{Q}(x, y) - p_l(x) P_l(x, y)$ so that $\deg_y \mathscr{Q}_{l-1} = l - 1$. Note that if $\deg_y \mathscr{Q} < l$, $p_l(x) = 0$. Since $(\mathscr{Q}, P_l) \in \mathscr{M}_l$, $\mathscr{Q}_{l-1} \in \mathscr{M}_l$. Continuing with $t = l - 1$, $P_{l-1}^{(l-1)}(x) = \prod_{j=0}^{n-1}(x - \alpha_j)^{\mathsf{m}_j(l-1)}$. Based on Lemma 3, $\prod_{j=0}^{n-1}(x - \alpha_j)^{\mathsf{m}_j(l-1)} \big| \mathscr{Q}_{l-1}^{(l-1)}(x)$. Therefore, $\mathscr{Q}_{l-2}(x, y)$ can be generated by $\mathscr{Q}_{l-2}(x, y) = \mathscr{Q}_{l-1}(x, y) - p_{l-1}(x) P_{l-1}(x, y)$ so that $\deg_y \mathscr{Q}_{l-2} = l - 2$. Following the above deduction until $t = 0$, $P_0^{(0)}(x) = \prod_{j=0}^{n-1}(x - \alpha_j)^{\mathsf{m}_j(0)}$ and $\prod_{j=0}^{n-1}(x - \alpha_j)^{\mathsf{m}_j(0)} \big| \mathscr{Q}_0^{(0)}(x)$. Hence, there exists $p_0(x)$ that enables $\mathscr{Q}_0(x, y) - p_0(x) P_0(x, y) = 0$.

```
Input: M, $l$;
Output: $\hat{f}(x)$;

1: Create balanced lists $L_0' \sim L_{n-1}'$;
2: Formulate $\mathcal{B}_l$ by (35) and map to $\mathcal{A}_l$ by (11);
3: Reduce $\mathcal{A}_l$ into $\mathcal{A}_l'$ and demap it to $\mathcal{B}_l'$ by (12);
4: Construct $Q$ by (13);
5: Find $y$-roots of $Q$ to estimate $\hat{f}(x)$.
```

Fig. 3 *Algorithm 3: the KV-MM algorithm*

Therefore, if $\mathscr{Q} \in \mathscr{M}_l$, it can be written as an $\mathbb{F}_q[x]$-linear combination of $P_t(x, y)$, i.e. $\mathscr{Q}(x, y) = \sum_{t=0}^{l} p_t(x) P_t(x, y)$. □

The above theorem reveals that (35) forms a basis $\mathscr{B}_l$ of $\mathscr{M}_l$. Presenting $\mathscr{B}_l$ as a matrix over $\mathbb{F}_q[x]$, $\mathscr{A}_l$ can be generated by (11). The MS algorithm will reduce $\mathscr{A}_l$ into $\mathscr{A}_l'$. Demap it by (12). Polynomial Q can be further retrieved.

The KV-MM algorithm is summarised in Fig. 3.

4.3 Re-encoding transformed KV-MM algorithm

Similar to the ACD-MM algorithm, re-encoding transform helps reduce degree of the basis entries, lowering the MM complexity. First, $\mathsf{m}_0(0), \mathsf{m}_1(0), \ldots, \mathsf{m}_{n-1}(0)$ are sorted to obtain an index sequence $j_0, j_1, \ldots, j_{n-1}$ such that $\mathsf{m}_{j_0}(0) \geq \mathsf{m}_{j_1}(0) \geq \cdots \geq \mathsf{m}_{j_{n-1}}(0)$. Let $\Upsilon = \{j_0, j_1, \ldots, j_{k-1}\}$ and $\bar{\Upsilon} = \{j_k, j_{k+1}, \ldots, j_{n-1}\}$. The k points $(\alpha_j, y_j^{(0)}), \forall j \in \Upsilon$ are chosen to form the re-encoding polynomial

$$H_\Upsilon(x) = \sum_{j \in \Upsilon} y_j^{(0)} \prod_{j' \in \Upsilon, j' \neq j} \frac{x - \alpha_{j'}}{\alpha_j - \alpha_{j'}}\,. \tag{36}$$

Therefore, $H_\Upsilon(\alpha_j) = y_j^{(0)}, \forall j \in \Upsilon$. All balanced lists L_j' will be transformed as

$$\tilde{L}_j' = [(\alpha_j, w_j^{(\varepsilon)}) \mid \varepsilon = 0, 1, \ldots, \mathsf{m}_j - 1], \tag{37}$$

where $w_j^{(\varepsilon)} = y_j^{(\varepsilon)} - H_\Upsilon(\alpha_j)$. For $j \in \Upsilon$, if $y_j^{(\varepsilon)} = y_j^{(0)}$, then $w_j^{(\varepsilon)} = 0$. Define $\Lambda_\varepsilon = \{j \mid w_j^{(\varepsilon)} = 0, j \in \Upsilon\}$ and $\bar{\Lambda}_\varepsilon = \Upsilon \setminus \Lambda_\varepsilon$. Note that $\Lambda_0 = \Upsilon$. Based on the transform, $F_\varepsilon(x)$ of (34) can be redefined as

$$F_\varepsilon(x) = \sum_{j=0}^{n-1} w_j^{(\varepsilon)} \Phi_j(x)\,. \tag{38}$$

Let us further define

$$\phi(x) = \prod_{j \in \Upsilon} (x - \alpha_j)^{\mathsf{m}_j(0)} \tag{39}$$

and

$$\psi(x) = \prod_{j \in \Upsilon} (x - \alpha_j)\,. \tag{40}$$

The following Lemma characterises module generators when re-encoding transform is applied.

Lemma 4: Given the multiplicity matrix $\boldsymbol{M}$ and the transformed lists of (37), $\phi(x) | P_t(x, y\psi(x))$ holds.

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

The following bijective mapping is further defined as

$$\begin{aligned} \varphi: \quad & \mathscr{M}_l \to \tilde{\mathscr{M}}_l \\ & P_t(x, y) \mapsto \phi(x)^{-1} P_t(x, y\psi(x))\,. \end{aligned} \tag{41}$$

Consequently, polynomials of $\tilde{\mathscr{M}}_l$ have the lower x-degree, leading to a simpler basis reduction. Let

$$T_\varepsilon(x) = \sum_{j \in \Upsilon \cup \bar{\Lambda}_\varepsilon} \tilde{w}_j^{(\varepsilon)} \prod_{j' \in \Upsilon \cup \bar{\Lambda}_\varepsilon, j' \neq j} (x - \alpha_{j'}), \tag{42}$$

where

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

the proof of Lemma 7 reveals that

$$\begin{aligned} P_t(x, y\psi(x)) = \phi(x) \cdot \prod_{j \in \Upsilon} (x - \alpha_j)^{\mathsf{m}_j(t)} \cdot \prod_{j \in \Lambda_t} (x - \alpha_j) \\ \times \prod_{\varepsilon = 0}^{t-1} \left(y \prod_{j \in \Lambda_\varepsilon} (x - \alpha_j) - T_\varepsilon(x) \right). \end{aligned} \tag{43}$$

Based on (41), $\tilde{\mathscr{M}}_l$ can be generated by

$$\begin{aligned} \tilde{P}_t(x, y) = \prod_{j \in \Upsilon} (x - \alpha_j)^{\mathsf{m}_j(t)} \cdot \prod_{j \in \Lambda_t} (x - \alpha_j) \\ \times \prod_{\varepsilon = 0}^{t-1} \left(y \prod_{j \in \Lambda_\varepsilon} (x - \alpha_j) - T_\varepsilon(x) \right), \end{aligned} \tag{44}$$

where $t = 0, 1, \ldots, l$. Note that $\Lambda_l = \varnothing$. The above polynomials $\tilde{P}_t(x, y)$ form a basis $\tilde{\mathscr{B}}_l$ of $\tilde{\mathscr{M}}_l$. Present $\tilde{\mathscr{B}}_l$ as a matrix over $\mathbb{F}_q[x]$ and perform the mapping of $\mathscr{A}_l = \tilde{\mathscr{B}}_l \cdot \tilde{\mathscr{D}}_{1,l}$. The MS algorithm will reduce $\mathscr{A}_l$ into $\mathscr{A}_l'$. Demap it by $\tilde{\mathscr{B}}_l' = \mathscr{A}_l' \cdot \tilde{\mathscr{D}}_{-1,l}$ and polynomial $\tilde{Q}$ can be further retrieved. Based on Lemma 4, the interpolated polynomial Q can be constructed by

$$Q(x, y) = \phi(x)\tilde{Q}\left(x, \frac{y}{\psi(x)}\right). \tag{45}$$

If $f'(x)$ is a y-root of Q, $\hat{f}(x)$ is estimated by $\hat{f}(x) = f'(x) + H_\Upsilon(x)$.

The re-encoding transformed KV-MM algorithm is summarised in Fig. 4

5 Complexity reducing approach

This section introduces a complexity reducing approach for the above mentioned algorithms. It is realised by assessing the degree of the module seeds.

The paradigm of the ACD-MM algorithm and its re-encoding transformed variant are first described.

Lemma 5: Given $R(x)$ that is defined as in (7), if $\underline{\omega} = (\omega_0, \omega_1, \ldots, \omega_{n-1})$ is a codeword, then $\deg R(x) < k$.

Proof: Based on (7), $\deg R(x) \leq n - 1$ and $R(\alpha_j) = \omega_j, \forall j$. If $\underline{\omega}$ is a codeword, there exists a message polynomial $\lambda(x) \in \mathbb{F}_q[x]$ with $\deg \lambda(x) < k$ such that $\lambda(\alpha_j) = \omega_j, \forall j$. Let

$$\lambda'(x) = R(x) - \lambda(x),$$

then $\deg\lambda'(x) \leq n - 1$. Since

```
Input: $\mathrm{M}, l$;
Output: $\hat{f}(x)$;

1: Create balanced lists $L_0' \sim L_{n-1}'$;
2: Sort $\mathsf{m}_0(0), \mathsf{m}_1(0), \ldots, \mathsf{m}_{n-1}(0)$ and define $\Upsilon$;
3: Transform all balanced lists as in (37);
4: Formulate $\tilde{\mathcal{B}}_l$ by (44) and map to $\mathcal{A}_l$ by (27);
5: Reduce $\mathcal{A}_l$ into $\mathcal{A}_l'$ and demap it to $\tilde{\mathcal{B}}_l'$ by (28);
6: Construct $Q$ by (13) and (45);
7: Find $y$-roots of $Q$ to estimate $\hat{f}(x)$.
```

Fig. 4 *Algorithm 4: the re-encoding transformed KV-MM algorithm*

$$\lambda'(\alpha_j) = R(\alpha_j) - \lambda(\alpha_j) = 0,$$

$\lambda'(x)$ has n roots. It can be written as

$$\lambda'(x) = \gamma(x) \cdot \prod_{j=0}^{n-1} (x - \alpha_j),$$

where $\gamma(x) \in \mathbb{F}_q[x]$. This leads to $\deg\lambda'(x) \geq n$, which contradicts to $\deg\lambda'(x) \leq n - 1$. Therefore, $\gamma(x) = 0$ and $\lambda'(x) = 0$. Hence, $R(x) = \lambda(x)$ and $\deg R(x) < k$. □

The above conclusion leads to the following Lemma.

Lemma 6: If $\deg R(x) < k$, $\mathscr{A}_l$ is in the weak Popov form.

Proof: When $0 \leq t \leq m$, based on (9) and (11)

$$\begin{aligned} \deg \mathscr{A}_l|_t^{(\tau)} &= n(m - t) + (t - \tau)\deg R(x) + (k - 1)\tau \\ &= nm - (n - \deg R(x))t + (k - 1 - \deg R(x))\tau. \end{aligned}$$

When $m < t \leq l$

$$\begin{aligned} \deg \mathscr{A}_l|_t^{(\tau)} &= (t - \tau)\deg R(x) + (k - 1)\tau \\ &= t\deg R(x) + (k - 1 - \deg R(x))\tau. \end{aligned}$$

Note that when $\tau > t$, $\mathscr{A}_l|_t^{(\tau)} = 0$. If $\deg R(x) < k$, then $\max\{\deg \mathscr{A}_l|_t^{(\tau)}, \forall\tau\} = \deg \mathscr{A}_l|_t^{(t)} = nm - (n - k + 1)t$ when $0 \leq t \leq m$, and $\max\{\deg \mathscr{A}_l|_t^{(\tau)}, \forall\tau\} = \deg \mathscr{A}_l|_t^{(t)} = (k - 1)t$ when $m < t \leq l$. Therefore, $\mathrm{LP}(\mathscr{A}_l|_t) = t$. Based on Definition 5, $\mathscr{A}_l$ is in the weak Popov form. □

The above two Lemmas reveal that if $\deg R(x) < k$, the constructed $\mathscr{B}_l$ is the desired Gröbner basis and $R(x)$ would be a message candidate. That says in the ACD-MM algorithm, if $\deg R_u(x) < k$, the test-vector $\underline{\boldsymbol{r}}_u$ is already a codeword. The maximum likelihood (ML) criterion [29] can be further applied to assess whether it is an ML codeword. If so, the decoding can be terminated and outputs $R_u(x)$ as a message candidate. The following basis construction and reduction can be skipped.

For an (n,k) RS code, given two distinct codewords $\underline{c}$ and $\underline{c}'$, $d_\mathrm{H}(\underline{c}, \underline{c}') \geq n - k + 1$. If $\eta \leq n - k$, for two distinct test-vectors $\underline{\boldsymbol{r}}_u$ and $\underline{\boldsymbol{r}}_{u'}$, $d_\mathrm{H}(\underline{\boldsymbol{r}}_u, \underline{\boldsymbol{r}}_{u'}) \leq \eta < d_\mathrm{H}(\underline{c}, \underline{c}')$. Hence, if $\underline{\boldsymbol{r}}_u$ is a codeword, then $\underline{\boldsymbol{r}}_{u'}$ will not be a codeword, $\forall u' \neq u$. Therefore, if $R_u(x)$ is output as the message, the other decoding trials can also be skipped. This would further reduce the complexity of the ACD-MM algorithm.

Similarly, for the re-encoding transformed ACD-MM algorithm, if $\deg R_u(x) < k$, $\underline{z}_u$ would be a codeword. The original test-vector $\underline{\boldsymbol{r}}_u$ should be further recovered by $r_j^{(u)} = z_j^{(u)} + H_\Theta(\alpha_j), \forall j$, and seen if it is an ML codeword. Therefore, in the ACD-MM algorithms (Algorithms 1 and 2 (see Figs. 1 and 2), once $\deg R_u(x) < k$, the decoding outputs $R_u(x)$ as a message candidate. If $\underline{\boldsymbol{r}}_u$ is an ML codeword, the decoding can be terminated.

This complexity reducing approach can also be applied to the KV-MM algorithms. Similar to the ACD-MM algorithm, if $\underline{y}_\varepsilon = (y_0^{(\varepsilon)}, y_1^{(\varepsilon)}, \ldots, y_{n-1}^{(\varepsilon)})$ is a codeword, $\deg F_\varepsilon(x) < k$. Subsequently, the following Lemma for the KV-MM algorithm can be obtained.

Lemma 7: If $\deg F_\varepsilon(x) < k, \forall\varepsilon$, $\mathscr{A}_l$ is in the weak Popov form.

Proof: Based on (35) and (11)

$$\begin{aligned} &\deg \mathscr{A}_l|_t^{(\tau)} \\ &= \sum_{j=0}^{n-1} \mathsf{m}_j(t) + \sum_{\varepsilon=0}^{t-\tau-1} \deg F_\varepsilon(x) + (k - 1)\tau \\ &= \sum_{j=0}^{n-1} \mathsf{m}_j(t) + \sum_{\varepsilon=0}^{t-1} \deg F_\varepsilon(x) + \sum_{\varepsilon=t-\tau}^{t-1} (k - 1 - \deg F_\varepsilon(x)). \end{aligned}$$

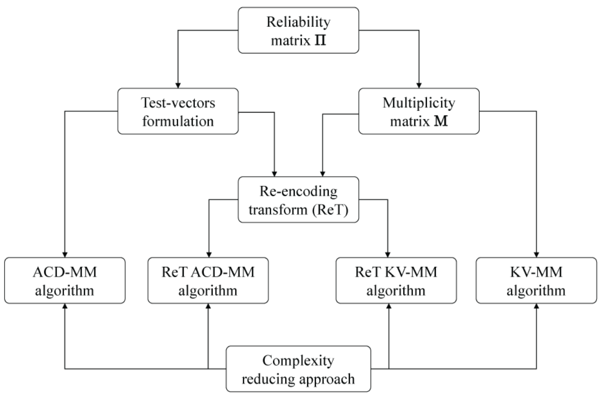

Fig. 5 *Block diagram of the proposed algorithms*

If $\deg F_\varepsilon(x) < k, \forall\varepsilon$, then $\max\{\deg\mathscr{A}_l|_t^{(\tau)}, \forall\tau\} = \deg\mathscr{A}_l|_t^{(t)} = \sum_{j=0}^{n-1}\mathsf{m}_j(t) + (k-1)t$, where $0 \le t \le l$. Therefore, $\mathrm{LP}(\mathscr{A}_l|_t) = t$ and $\mathscr{A}_l$ is in the weak Popov form. □

Consequently, in the KV-MM algorithm, if $\deg F_\varepsilon(x) < k, \forall\varepsilon$, $\mathscr{B}_l$ is the desired Gröbner basis and all $F_\varepsilon(x)$ are message candidates. The decoding can further apply the ML criterion to validate an ML codeword and output the corresponding $F_\varepsilon(x)$ as the message. For the re-encoding transformed variant, if $\deg F_\varepsilon(x) < k, \forall\varepsilon$, all transformed vectors $\boldsymbol{w}_\varepsilon = (w_0^{(\varepsilon)}, w_1^{(\varepsilon)}, \dots, w_{n-1}^{(\varepsilon)})$ are codewords. After recovering $\underline{y}_\varepsilon$ by $y_j^{(\varepsilon)} = w_j^{(\varepsilon)} + H_\Upsilon(\alpha_j), \forall j$, an ML codeword can again be validated. Once an ML codeword has been identified, the decoding will output the corresponding polynomial $F_\varepsilon(x)$ as the message and terminate afterwards.

Summarising the descriptions of Sections 3–5, Fig. 5 shows an overview of the proposed algorithms.

6 Complexity analysis

This section analyses complexity of the MM interpolation that consists of basis construction and reduction. It dominates complexity of the two ASD algorithms and their re-encoding transformed variants. In this paper, the complexity refers to the number of finite field operations in decoding a codeword.

6.1 Without the re-encoding transform

Complexity of the basis construction is first analysed. For the ACD-MM algorithm, $G(x)$ and $\Phi_j(x)$ can be computed offline. Computing $R(x)$ requires n^2 operations. Complexity of constructing $\mathscr{B}_l$ is $\sum_{t=0}^{m}\sum_{j=0}^{t}(t-j)(n-1)\cdot(m-t)n \simeq \frac{1}{24}n^2(m+1)^4$ by using the naive polynomial multiplication. Therefore, in the ACD-MM algorithm, with η unreliable symbols, complexity of the basis construction is $2^\eta(\frac{1}{24}n^2((m+1)^4+24))$. For the KV-MM algorithm, the construction of $F_\varepsilon(x)$ and $\mathscr{B}_l$ require $n^2 1$ and $\frac{1}{24}n^2(l+1)^4$ operations, respectively. Hence, the basis construction complexity is $\frac{1}{24}n^2((l+1)^4+24l)$.

Complexity of the basis reduction will be determined by the degree of $\mathscr{A}_l|_t^{(\tau)}$ and the number of row operations for reducing $\mathscr{A}_l$ into $\mathscr{A}_l'$.

Lemma 8: Given a matrix $\mathscr{A}_l$ over $\mathbb{F}_q[x]$, there are less than $l(\deg\mathscr{A}_l - \deg\det\mathscr{A}_l + l)$ row operations to reduce it into the weak Popov form $\mathscr{A}_l'$ [28].

The following two Lemmas further characterise $\deg\mathscr{A}_l|_t^{(\tau)}$ and $\deg\mathscr{A}_l - \deg\det\mathscr{A}_l$, respectively.

Lemma 9: Without the re-encoding transform, $\deg\mathscr{A}_l|_t^{(\tau)} \le nl$.

Proof: For the ACD-MM algorithm, $\deg\mathscr{A}_l|_t^{(\tau)}$ is determined based on the generators (9) and (10) and mapping (11). When $0 \le t \le m$

$$\begin{aligned}\deg\mathscr{A}_l|_t^{(\tau)} &= n(m-t) + (n-1)(t-\tau) + (k-1)\tau\\ &= nm - t - (n-k)\tau\,.\end{aligned}$$

Therefore, $\max\{\deg\mathscr{A}_l|_t^{(\tau)}, 0 \le t \le m\} = \deg\mathscr{A}_l|_0^{(0)} = nm$. When $m < t \le l$

$$\begin{aligned}\deg\mathscr{A}_l|_t^{(\tau)} &= (n-1)(t-\tau) + (k-1)\tau\\ &= (n-1)t - (n-k)\tau,\end{aligned}$$

and $\max\{\deg\mathscr{A}_l|_t^{(\tau)}, m < t \le l\} = \deg\mathscr{A}_l|_l^{(0)} = (n-1)l$. Therefore, for the ACD-MM algorithm, $\max\{\deg\mathscr{A}_l|_t^{(\tau)}, \forall(t,\tau)\} = \max\{nm, (n-1)l\} \le nl$.

For the KV-MM algorithm, the generators (35) and mapping (11) lead to

$$\begin{aligned}\deg\mathscr{A}_l|_t^{(\tau)} &\le n(l-t) + (n-1)(t-\tau) + (k-1)\tau\\ &= nl - t - (n-k)\tau\,.\end{aligned}$$

Hence, $\max\{\deg\mathscr{A}_l|_t^{(\tau)}, \forall(t,\tau)\} = \deg\mathscr{A}_l|_0^{(0)} = nl$. □

Lemma 10: Without the re-encoding transform, $\deg\mathscr{A}_l - \deg\det\mathscr{A}_l \le \frac{1}{2}(n-k)(l^2+l)$.

Proof: For the ACD-MM algorithm, when $0 \le t \le m$, $\deg\mathscr{A}_l|_t = nm - t$. When $m < t \le l$, $\deg\mathscr{A}_l|_t = (n-1)t$. Therefore

$$\deg\mathscr{A}_l = \sum_{t=0}^{m}(nm-t) + \sum_{t=m+1}^{l}(n-1)t$$

and

$$\deg\det\mathscr{A}_l = \sum_{t=0}^{m}n(m-t) + \sum_{t=0}^{l}(k-1)t\,.$$

Hence, $\deg\mathscr{A}_l - \deg\det\mathscr{A}_l = \sum_{t=0}^{l}(n-k)t = \frac{1}{2}(n-k)(l^2+l)$.

For the KV-MM algorithm, let $\tau_t = \mathrm{LP}(\mathscr{A}_l|_t)$, then

$$\deg\mathscr{A}_l = \sum_{t=0}^{l}\left(\sum_{j=0}^{n-1}\mathsf{m}_j(t) + (n-1)(t-\tau_t) + (k-1)\tau_t\right).$$

Since $\mathscr{A}_l$ is a lower-triangle matrix and $P_t^{(t)}(x) = 1$

$$\deg\det\mathscr{A}_l = \sum_{t=0}^{l}\left(\sum_{j=0}^{n-1}\mathsf{m}_j(t) + (k-1)t\right).$$

Therefore, $\deg\mathscr{A}_l - \deg\det\mathscr{A}_l \le \sum_{t=0}^{l}((n-1)(t-\tau_t) + (k-1)\tau_t - (k-1)t)$. When $\tau_t = 0$, $\max\{\deg\mathscr{A}_l - \deg\det\mathscr{A}_l\} = \frac{1}{2}(n-k)(l^2+l)$. □

Based on the above lemmas, it can be concluded that the basis reduction process requires at most $\frac{1}{2}n(n-k)l^3(l+1)^2$ finite field operations.

Therefore, when l is sufficiently large, with η unreliable symbols, the MM complexity is $O(2^\eta n(n-k)l^5)$ in the ACD algorithm. Meanwhile, the MM complexity is $O(n(n-k)l^5)$ in the KV algorithm. These conclusions imply the MM complexity favours high rate codes, which is of practical interest. Table 1 shows the numerical results of the MM complexity in decoding various RS codes defined over $\mathbb{F}_{64}$. On one hand, they verify the above complexity characterisations. On the other hand, they show a

Table 1 Interpolation complexity of the ACD-MM and KV-MM algorithms

		(63, 31) RS	(63, 47) RS	(63, 55) RS
ACD-MM	$(m, l, \eta) = (1, 1, 3)$	1.16×10^5	9.96×10^4	9.22×10^4
	$(m, l, \eta) = (5, 7, 3)$	5.56×10^7	—	—
KV-MM	$l = 4$	1.68×10^6	1.22×10^6	1.01×10^6
	$l = 8$	2.84×10^7	1.95×10^7	1.45×10^7

Table 2 Interpolation complexity of the re-encoding transformed ACD-MM and KV-MM algorithms

		(63, 31) RS	(63, 47) RS	(63, 55) RS
ACD-MM	$(m, l, \eta) = (1, 1, 3)$	7.72×10^4	3.01×10^4	1.41×10^4
	$(m, l, \eta) = (5, 7, 3)$	3.50×10^7	—	—
KV-MM	$l = 4$	1.13×10^6	3.05×10^5	1.52×10^5
	$l = 8$	8.62×10^6	1.91×10^6	8.47×10^5

high rate code yields a lower MM complexity. Table 1 also shows that with the same decoding output cardinality, e.g. the ACD-MM $(m = 1, l = 1, \eta = 3)$ and the KV-MM $(l = 8)$, the ACD-MM algorithm is less complex by at least two orders of magnitude.

6.2 With the re-encoding transform

Re-encoding transform reduces the degree of module generators, leading to a simpler basis reduction. Complexity of formulating the re-encoding polynomial and the module basis $\tilde{\mathscr{B}}_l$ are first characterised. To compute $H_\Theta(x)$ (or $H_\Upsilon(x)$), $3(n-k)^2$ finite field operations are required [9]. The following interpolation point transform requires $(n-k)k$ operations. In the ACD-MM algorithm, computing $\tilde{G}(x)$ and $\tilde{R}_u(x)$ require $\frac{1}{2}(n-k)^2$ and $\frac{1}{2}(n-k)^2(n-k-1) + (n-k)(n-k-1) + n(n-k) \simeq \frac{1}{2}(n-k)^3$ operations, respectively. Further constructing $\tilde{\mathscr{B}}_l$ requires $\sum_{t=0}^{m}\sum_{j=0}^{t}(t-j)(n-k-1)\cdot(m-t)(n-k) \simeq \frac{1}{24}(n-k)^2(m+1)^4$ operations. Therefore, the basis construction complexity for each test-vector is $\frac{1}{24}(n-k)^2((m+1)^4 + 12(n-k))$. For the KV-MM algorithm, computing $T_\varepsilon(x)$ and $\tilde{\mathscr{B}}_l$ require $\frac{1}{2}(n-k)^3$ and $\frac{1}{24}(n-k)^2(l+1)^4$ finite field operations, respectively. Note that it is assumed $\bar{\Lambda}_\varepsilon = \varnothing, \forall\varepsilon$. Therefore, complexity of the basis construction is $\frac{1}{24}(n-k)^2((l+1)^4 + 12(n-k))$.

To characterise complexity of the basis reduction, the following two Lemmas are needed.

Lemma 11: With the re-encoding transform, $\deg\mathscr{A}_l|_t^{(\tau)} \le (n-k+1)l$.

Proof: For the ACD-MM algorithm, when $0 \le t \le m$

$$\deg\mathscr{A}_l|_t^{(\tau)} = (n-k)(m-t) + (n-k-1)(t-\tau) + (l-\tau)\,.$$

Since $m \le l$, $\max\{\deg\mathscr{A}_l|_t^{(\tau)}, 0 \le t \le m\} = \deg\mathscr{A}_l|_0^{(0)} \le (n-k+1)l$. When $m < t \le l$

$$\deg\mathscr{A}_l|_t^{(\tau)} = k(t-m) + (n-k-1)(t-\tau) + (l-\tau),$$

and $\max\{\deg\mathscr{A}_l|_t^{(\tau)}, m < t \le l\} = \deg\mathscr{A}_l|_l^{(0)} = (n-k)l$.

With the re-encoding transform, entry size of $\mathscr{A}_l$ in the KV-MM algorithm is

$$\deg\mathscr{A}_l|_t^{(\tau)} \le (n-k)(l-t) + (n-k-1)(t-\tau) + (l-\tau)\,.$$

Hence, $\max\{\deg\mathscr{A}_l|_t^{(\tau)}, \forall(t,\tau)\} = \deg\mathscr{A}_l|_0^{(0)} = (n-k+1)l$. □

Lemma 12: With the re-encoding transform, $\deg\mathscr{A}_l - \deg\det\mathscr{A}_l \le \frac{1}{2}(n-k)(l^2+l)$.

Proof: For the ACD-MM algorithm

$$\deg\mathscr{A}_l = \sum_{t=0}^{m}((n-k)m - t + l) + \sum_{t=m+1}^{l}((n-1)t + (l-km))$$

and

$$\deg\det\mathscr{A}_l$$
$$= \sum_{t=0}^{m}((n-k)(m-t) + (l-t)) + \sum_{t=m+1}^{l}(k(t-m) + (l-t))\,.$$

Hence, $\deg\mathscr{A}_l - \deg\det\mathscr{A}_l = \sum_{t=0}^{l}(n-k)t = \frac{1}{2}(n-k)(l^2+l)$.

For the KV-MM algorithm, let $\tau_t = \mathrm{LP}(\mathscr{A}_l|_t)$, then

$$\deg\mathscr{A}_l - \deg\det\mathscr{A}_l$$
$$\le \sum_{t=0}^{l}((n-k+1)(t-\tau_t) + (l-\tau_t) - (l-t))\,.$$

When $\tau_t = 0$, $\max\{\deg\mathscr{A}_l - \deg\det\mathscr{A}_l\} = \frac{1}{2}(n-k)(l^2+l)$. □

Therefore, echoing Lemma 10, the re-encoding transform does not attribute to reducing the number of row operations during the basis reduction process. Further applying Lemma 8, with the re-encoding transform, the basis reduction requires at most $\frac{1}{2}(n-k)^2 l^3(l+1)^2$ finite field operations. Therefore, for the re-encoding transformed variants, the MM complexity would be $O(2^\eta(n-k)^2 l^5)$ and $O((n-k)^2 l^5)$ in the ACD and the KV algorithms, respectively.

Compared with the analysis of Section 6.1, re-encoding transform can help reduce the MM complexity by a factor of k/n. Therefore, re-encoding transform is more effective in reducing complexity for high rate codes. Table 2 shows the numerical results of the MM complexity when applying the re-encoding transform, verifying the above analysis. It again shows with the same decoding output cardinality, the ACD-MM algorithm yields a much lower complexity. By comparing Tables 1 and 2, the re-encoding transform can also reduce the MM complexity by a factor of k/n.

7 Simulation results

This section presents simulation results of the ACD and the KV algorithms. They are obtained over the additive white Gaussian noise channel using BPSK modulation. Their competency in terms of decoding performance and complexity will be shown, giving more practical insights. Again, the decoding complexity is measured as the number of finite field operations that is required to decode a codeword, including the re-encoding transform and the root-finding. The ACD and the KV algorithms that employ Koetter's interpolation are denoted as the ACD-Koetter and the KV-Koetter algorithms, respectively.

7.1 Performance comparison

Figs. 6 and 7 show the ACD and the KV performance in decoding the (63, 31) RS code, respectively, where the decoding performance is measured by frame error rate. Both of the ASD algorithms outperform the BM algorithm thanks to their soft decoding feature. Fig. 6 shows when $m = 1$, the ACD algorithm cannot outperform the GMD algorithm. But when $m = 5$, each Chase decoding trial can correct at most 18 symbol errors. As a result, the ACD performance is better than the GMD. Meanwhile, Fig. 7 shows the KV algorithm outperforms the GMD algorithm when $l \ge 2$. By comparing Figs. 6 and 7, when $m = 1$, the ACD algorithm with $\eta = 2, 3, 4$ yields the same decoding output

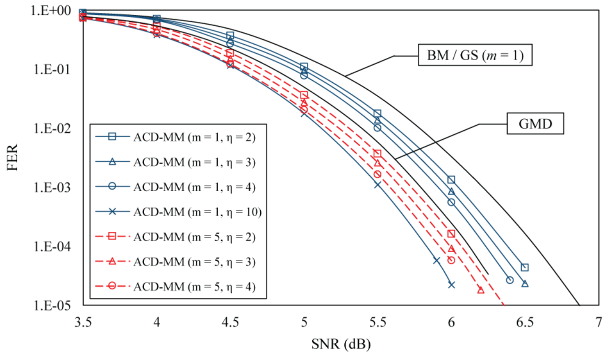

Fig. 6 *Performance of the ACD algorithm in decoding the (63, 31) RS code*

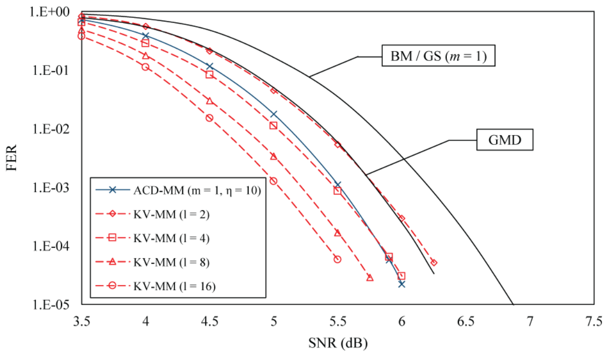

Fig. 7 *Performance of the KV algorithm in decoding the (63, 31) RS code*

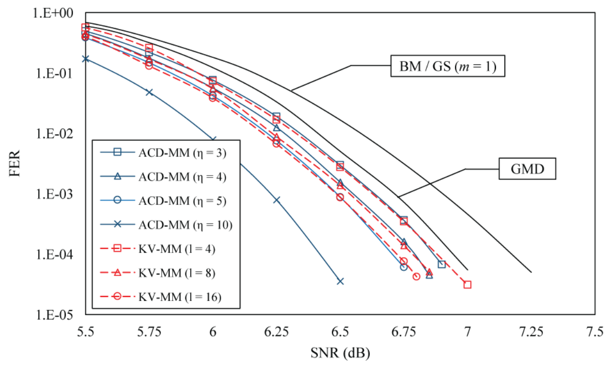

Fig. 8 *Performance of the (255, 239) RS code*

Table 3 Complexity of the ACD algorithm in decoding the (63, 31) RS code

(m, l, η)	W/o re-encoding		With re-encoding	
	ACD-MM	ACD-Koetter	ACD-MM	ACD-Koetter
(1, 1, 3)	2.77×10^5	3.30×10^5	2.11×10^5	1.71×10^5
(5, 7, 3)	6.22×10^7	1.97×10^8	3.83×10^7	7.43×10^7
(1, 1, 10)	3.01×10^7	2.63×10^7	1.37×10^7	1.16×10^7

Table 4 Complexity of the KV algorithm in decoding the (63, 31) RS code

l	W/o re-encoding		With re-encoding	
	KV-MM	KV-Koetter	KV-MM	KV-Koetter
4	1.82×10^6	1.59×10^7	1.25×10^6	6.16×10^6
8	3.01×10^7	3.50×10^8	9.84×10^6	1.10×10^8

cardinality as the KV algorithm with $l = 4, 8, 16$, respectively. Under such a benchmark, the KV algorithm yields a better decoding performance.

Fig. 8 compares the two ASD algorithms in decoding the popular (255, 239) RS code. Note that the ACD algorithm

Table 5 Complexity of the re-encoding transformed ACD and KV algorithms in decoding the (255, 239) RS code

(m, l, η)	ACD-MM	ACD-Koetter	BF-ACD [10]	l	KV-MM	KV-Koetter
(1, 1, 3)	1.92×10^6	2.58×10^6	2.82×10^6	4	6.54×10^6	1.06×10^8
(1, 1, 4)	3.77×10^6	4.65×10^6	5.49×10^6	8	4.27×10^7	3.24×10^8

performs with $m = 1$. For this high rate code, both of the ASD algorithms outperform the BM and the GMD algorithms. With the same decoding output cardinality, e.g. the ACD ($\eta = 3$) and the KV ($l = 8$), and the ACD ($\eta = 4$) and the KV ($l = 16$), the KV algorithm still outperforms the ACD algorithm.

7.2 Complexity comparison

Tables 3 and 4 show the ACD and the KV complexity in decoding the (63, 31) RS code, respectively. Complexity of the ACD-MM and the ACD-Koetter algorithms are compared first. When $m = 1$, the two interpolation complexity are similar. However, the ACD-MM algorithm can decode the test-vectors in parallel, leveraging the decoding latency to that of a single decoding event. This is an advantage over the ACD-Koetter algorithm [9] whose low complexity is forged in a binary tree growth fashion. When $m = 5$, the MM interpolation yields a much lower complexity than Koetter's interpolation despite whether the re-encoding transform is applied. For the KV algorithm, the MM interpolation also exhibits a lower complexity than Koetter's interpolation. Note that Fig. 7 shows performance of the ACD ($m = 1, \eta = 10$) is similar to that of the KV ($l = 4$). Tables 3 and 4 show that the KV-MM algorithm is less complex, despite whether the re-encoding transform is applied. On the other hand, with a similar decoding complexity, e.g. the ACD-MM ($m = 1, \eta = 10$) and the KV-MM ($l = 8$), the KV decoding outperforms.

Table 5 further shows complexity of the re-encoding transformed ACD and KV algorithms in decoding the (255, 239) RS code. It shows that MM interpolation yields a lower complexity for both of the ASD algorithms. Recalling Fig. 8, the ACD with $\eta = 3, 4$ performs similarly as the KV with $l = 4, 8$, respectively. Table 5 shows the ACD algorithm is less complex using both of the interpolation techniques. Hence, for high rate codes, the ACD algorithm becomes a better option in practice. Moreover, complexity of the BF-ACD algorithm [10] is shown in Table 5. Note that it also applies Koetter's interpolation. With the same η value, the BF-ACD algorithm is slightly more complex than the other two ACD algorithms. This extra computational cost is due to the backward interpolation of the BF-ACD algorithm, in which the polynomial size is always greater than that of the others.

7.3 Effectiveness of complexity reduction

Table 6 shows the reduced complexity of the ACD-MM algorithms yielded by the proposed approach of Section 5. It can be seen that as the SNR increases, the complexity reducing approach becomes more effective. This is because more decoding events would yield $\deg R_u(x) < k$ and $\underline{r}_u$ also satisfies the ML criterion. In decoding the (63, 31) RS code, the research shows when SNR = 8 dB, 26% of the decoding events are terminated by the above criterion. When SNR = 9 dB, this portion raises to 74%. Consequently, the complexity can be reduced by at most an order of magnitude. A similar phenomenon can also be observed for the case of using the re-encoding transform.

Table 7 further shows the reduced complexity of the re-encoding transformed KV-MM algorithm with $l = 4$. It again shows a significant complexity reduction at high SNR. Moreover, Tables 6 and 7 also show for the original ACD-MM and the KV-MM algorithms, their complexity also decreases with the SNR. This is due to the fact that in the high SNR region, many of the constructed basis would already be the desired Gröbner basis (they are already in the weak Popov form). The following basis reduction can be skipped.

Table 6 Reduced complexity of the ACD-MM algorithms in decoding the (63, 31) RS code

$\eta = 3$	SNR, dB	4	5	6	7	8	9	10
$m = 1, l = 1$	original	2.77×10^5	2.73×10^5	2.65×10^5	2.57×10^5	2.52×10^5	2.50×10^5	2.49×10^5
w/o re-enc.	reduced	2.77×10^5	2.73×10^5	2.64×10^5	2.50×10^5	1.94×10^5	7.92×10^4	1.98×10^4
$m = 5, l = 7$	original	6.22×10^7	5.68×10^7	4.59×10^7	4.13×10^7	3.63×10^7	3.18×10^7	3.01×10^7
w/o re-enc.	reduced	6.20×10^7	5.66×10^7	4.57×10^7	4.04×10^7	2.88×10^7	9.87×10^6	1.33×10^6
$m = 5, l = 7$	original	3.83×10^7	3.19×10^7	2.07×10^7	1.43×10^7	1.12×10^7	9.44×10^6	8.82×10^6
with re-enc.	reduced	3.83×10^7	3.18×10^7	2.05×10^7	1.41×10^7	9.07×10^6	3.08×10^6	4.54×10^5

Table 7 Reduced complexity of the re-encoding transformed KV-MM algorithm

$l = 4$	SNR, dB	4	5	6	7	8	9	10
(63, 55)	original	7.51×10^5	6.26×10^5	2.81×10^5	1.88×10^5	1.81×10^5	1.79×10^5	1.79×10^5
	reduced	7.50×10^5	5.77×10^5	1.84×10^5	4.99×10^4	1.72×10^4	5.54×10^3	2.73×10^3
(255, 239)	original	6.54×10^6	5.99×10^6	5.61×10^6	2.61×10^6	2.32×10^6	2.29×10^6	2.29×10^6
	reduced	6.54×10^6	5.99×10^6	5.54×10^6	1.90×10^6	6.08×10^5	1.58×10^5	4.27×10^4

8 Conclusions

This paper has presented the low-complexity ACD-MM and KV-MM algorithms for RS codes. Their interpolation are realised by the MM technique which constructs a module basis and further reduces it into the Gröbner basis. Re-encoding transformed variants of the two ASD algorithms have also been presented. They yield a smaller MM complexity by reducing the degree of module generators. Moreover, a complexity reducing approach has been proposed for the ASD algorithms based on assessing the degree of module seeds. Complexity analysis has shown that both the MM interpolation and the re-encoding transform are more effective in yielding a low complexity for high rate codes. These results fall into the interest of practice where high rate codes are favoured. Simulation results have verified that the MM interpolation enables a significant complexity reduction for the two ASD algorithms, despite whether the re-encoding transform is applied. A comprehensive decoding performance and complexity comparison between the two ASD algorithms has also been conducted. For medium rate codes, the KV algorithm outperforms the ACD algorithm under a similar decoding expenditure. While for high rate codes, the ACD algorithm prevails. Finally, numerical results have demonstrated the effectiveness of the proposed complexity reducing approach.

9 Acknowledgments

This work is sponsored by the National Natural Science Foundation of China (NSFC) with project ID 61671486 and International Program for PhD candidates, Sun Yat-sen University.

10 References

[1] Berlekamp, E.: '*Algebraic coding theory*' (McGraw-Hill, New York, 1968)

[2] Massey, J.: 'Shift register synthesis and BCH decoding', *IEEE Trans. Inf. Theory*, 1969, **15**, (1), pp. 122–127

[3] Kotter, R.: 'Fast generalized minimum-distance decoding of algebraic-geometry and Reed-Solomon codes', *IEEE Trans. Inf. Theory*, 1996, **42**, (3), pp. 721–737

[4] Sudan, M.: 'Decoding of Reed-Solomon codes beyond the error-correction bound', *J. Complexity*, 1997, **13**, (1), pp. 180–193

[5] Guruswami, V., Sudan, M.: 'Improved decoding of Reed-Solomon and algebraic-geometric codes', *IEEE Trans. Inf. Theory*, 1999, **45**, (6), pp. 1757–1767

[6] Koetter, R: '*On algebraic decoding of algebraic-geometric and cyclic codes*' (Linköping University, Linköping, Sweden, 1996)

[7] Nielsen, R., Høholdt, T.: 'Decoding Reed-Solomon codes beyond half the minimum distance', *Coding Theory, Cryptography Related Areas*, 2000, pp. 221–236

[8] Koetter, R., Vardy, A.: 'Algebraic soft-decision decoding of Reed-Solomon codes', *IEEE Trans. Inf. Theory*, 2003, **49**, (11), pp. 2809–2825

[9] Bellorado, J., Kavčić, A.: 'Low-complexity soft-decoding algorithms for Reed-Solomon codes – part I: an algebraic soft-in hard-out chase decoder', *IEEE Trans. Inf. Theory*, 2010, **56**, (3), pp. 945–959

[10] Zhang, X., Zheng, Y.: 'Generalized backward interpolation for algebraic soft-decision decoding of Reed-Solomon codes', *IEEE Trans. Commun.*, 2013, **61**, (1), pp. 13–23

[11] Wu, Y.: 'New list decoding algorithms for Reed-Solomon and BCH codes', *IEEE Trans. Inf. Theory*, 2008, **54**, (8), pp. 3611–3630

[12] Koetter, R., Vardy, A: 'A complexity reducing transformation in algebraic list decoding of Reed-Solomon codes'. Proc. IEEE Information Theory Workshop (ITW), Paris, France, 2003, pp. 10–13

[13] Koetter, R., Ma, J., Vardy, A.: 'The re-encoding transformation in algebraic list-decoding of Reed-Solomon codes', *IEEE Trans. Inf. Theory*, 2011, **57**, (2), pp. 633–647

[14] Chen, L., Tang, S., Ma, X.: 'Progressive algebraic soft-decision decoding of Reed-Solomon codes', *IEEE Trans. Commun.*, 2013, **61**, (2), pp. 433–442

[15] O'Keeffe, H., Fitzpatrick, P.: 'Gröbner basis solutions of constrained interpolation problems', *Linear Algebr. Appl.*, 2002, **351**, pp. 533–551

[16] Lee, K., O'Sullivan, M.: 'List decoding of Reed-Solomon codes from a Gröbner basis perspective', *J. Symb. Comput.*, 2008, **43**, (9), pp. 645–658

[17] Lee, K., O'Sullivan, M: 'An interpolation algorithm using Gröbner bases for soft-decision decoding of Reed-Solomon codes'. Proc. IEEE Int. Symp. Inform. Theory (ISIT), Seattle, USA, 2006, pp. 2032–2036

[18] Ma, J., Vardy, A.: 'A complexity reducing transformation for the Lee-O'Sullivan interpolation algorithm'. Proc. IEEE Int. Symp. Information Theory (ISIT), Nice, France, 2007, pp. 1986–1990

[19] Mulders, T., Storjohann, A.: 'On lattice reduction for polynomial matrices', *J Symb. Comput*, 2003, **35**, (4), pp. 377–401

[20] Alekhnovich, M.: 'Linear Diophantine equations over polynomials and soft decoding of Reed-Solomon codes', *IEEE Trans. Inf. Theory*, 2005, **51**, (7), pp. 2257–2265

[21] Cohn, H., Heninger, N.: 'Ideal forms of Coppersmith's theorem and Guruswami-Sudan list decoding', *Adv. Math. Commun.*, 2015, **9**, (3), pp. 311–339

[22] Jeannerod, C.P., Neiger, V., Schost, É., *et al.*: 'Computing minimal interpolation bases', *J. Symb. Comput.*, 2017, **83**, pp. 272–314

[23] Beelen, P., Høholdt, T., Nielsen, J., *et al.*: 'On rational interpolation-based list-decoding and list-decoding binary Goppa codes', *IEEE Trans. Inf. Theory*, 2013, **59**, (6), pp. 3269–3281

[24] Nielsen, J.S.R., Zeh, A.: 'Multi-trial Guruswami-Sudan decoding for generalised Reed-Solomon codes', *Des. Codes Cryptogr.*, 2014, **73**, (2), pp. 507–527

[25] Chen, L., Bossert, M.: 'Algebraic Chase decoding of Reed-Solomon codes using module minimisation'. Proc. Int. Symp. Information Theory Applications (ISITA), Monterey, USA, 2016, pp. 310–314

[26] Nielsen, J.: 'Power decoding Reed-Solomon codes up to the Johnson radius', *Adv. Math. Commun.*, 2018, **12**, (1), pp. 81–106

[27] Roth, R., Ruckenstein, G.: 'Efficient decoding of Reed-Solomon codes beyond half the minimum distance', *IEEE Trans. Inf. Theory*, 2000, **46**, (1), pp. 246–257

[28] Nielsen, J.: 'List decoding of algebraic codes'. Technical University of Denmark, Denmark,, 2013

[29] Kaneko, T., Nishijima, T., Inazumi, H., *et al.*: 'An efficient maximum-likelihood-decoding algorithm for linear block codes with algebraic decoder', *IEEE Trans. Inf. Theory*, 1994, **40**, (2), pp. 320–327

11 Appendix: Proof of Lemma 7

Proof: With the module generators defined by (35), it holds

$$\begin{aligned} &P_t(x, y\psi(x)) \\ &= \prod_{j=0}^{n-1} (x-\alpha_j)^{\mathsf{m}_j(t)} \prod_{\varepsilon=0}^{t-1} (y\psi(x) - F_\varepsilon(x)) \\ &= \prod_{j \in \Upsilon} (x-\alpha_j)^{\mathsf{m}_j(t)} \prod_{j \in \bar{\Upsilon}} (x-\alpha_j)^{\mathsf{m}_j(t)} \prod_{\varepsilon=0}^{t-1} (y\psi(x) - F_\varepsilon(x)) . \end{aligned} \tag{46}$$

Based on $F_\varepsilon(x)$ of (38) and the fact that $w_j^{(\varepsilon)} = 0, \forall j \in \Lambda_\varepsilon$ (see (47)) , where

$$T_\varepsilon(x) = \sum_{j \in \bar{\Upsilon} \cup \bar{\Lambda}_\varepsilon} \tilde{w}_j^{(\varepsilon)} \prod_{j' \in \bar{\Upsilon} \cup \bar{\Lambda}_\varepsilon, j' \neq j} (x - \alpha_{j'})$$

and

$$\tilde{w}_j^{(\varepsilon)} = \frac{w_j^{(\varepsilon)}}{\prod_{j'=0, j' \neq j}^{n-1} (\alpha_j - \alpha_{j'})} .$$

An equivalent expression of $\prod_{\varepsilon=0}^{t-1} \prod_{j \in \Lambda_\varepsilon} (x-\alpha_j)$ can now be derived. When $\varepsilon = 0$, $\Lambda_0 = \Upsilon$ and $\prod_{j \in \Lambda_0} (x-\alpha_j) = \prod_{j \in \Upsilon} (x-\alpha_j)$. Armed with the transformed lists of (37), when $\varepsilon \geq 1$, if $j \in \Lambda_\varepsilon$, $\mathsf{m}_j(\varepsilon-1) - \mathsf{m}_j(\varepsilon) = 1$. Otherwise, $\mathsf{m}_j(\varepsilon-1) = \mathsf{m}_j(\varepsilon)$. Therefore

$$\begin{aligned} &\prod_{\varepsilon=0}^{t-1} \prod_{j \in \Lambda_\varepsilon} (x-\alpha_j) \\ &\quad = \prod_{j \in \Upsilon} (x-\alpha_j) \cdot \prod_{\varepsilon=1}^{t-1} \prod_{j \in \Upsilon} (x-\alpha_j)^{\mathsf{m}_j(\varepsilon-1) - \mathsf{m}_j(\varepsilon)} \\ &\quad = \phi(x) \cdot \prod_{j \in \Upsilon} (x-\alpha_j)^{1 - \mathsf{m}_j(t-1)} . \end{aligned} \tag{48}$$

where $\phi(x) = \prod_{j \in \Upsilon} (x-\alpha_j)^{\mathsf{m}_j(0)}$. Based on (46)–(48)

$$\begin{aligned} P_t(x, y\psi(x)) = \phi(x) \cdot \prod_{j \in \Upsilon} (x-\alpha_j)^{\mathsf{m}_j(t)} \cdot U_t(x) \\ \times \prod_{\varepsilon=0}^{t-1} \left(y \prod_{j \in \Lambda_\varepsilon} (x-\alpha_j) - T_\varepsilon(x) \right), \end{aligned}$$

where $U_t(x) = \prod_{j \in \Upsilon} (x-\alpha_j)^{1 - (\mathsf{m}_j(t-1) - \mathsf{m}_j(t))} = \prod_{j \in \bar{\Lambda}_t} (x-\alpha_j)$. As a result

$$\begin{aligned} P_t(x, y\psi(x)) = \phi(x) \cdot \prod_{j \in \Upsilon} (x-\alpha_j)^{\mathsf{m}_j(t)} \cdot \prod_{j \in \Lambda_t} (x-\alpha_j) \\ \times \prod_{\varepsilon=0}^{t-1} \left(y \prod_{j \in \Lambda_\varepsilon} (x-\alpha_j) - T_\varepsilon(x) \right). \end{aligned}$$

□

$$\begin{aligned} &\prod_{\varepsilon=0}^{t-1} (y\psi(x) - F_\varepsilon(x)) \\ &\quad = \prod_{\varepsilon=0}^{t-1} \left(y\psi(x) - \sum_{j=0}^{n-1} w_j^{(\varepsilon)} \prod_{j'=0, j' \neq j}^{n-1} \frac{x - \alpha_{j'}}{\alpha_j - \alpha_{j'}} \right) \\ &\quad = \prod_{\varepsilon=0}^{t-1} \left(y \prod_{j \in \Lambda_\varepsilon} (x-\alpha_j) \prod_{j \in \bar{\Lambda}_\varepsilon} (x-\alpha_j) - \prod_{j \in \bar{\Lambda}_\varepsilon} (x-\alpha_j) \cdot T_\varepsilon(x) \right) \\ &\quad = \prod_{\varepsilon=0}^{t-1} \prod_{j \in \bar{\Lambda}_\varepsilon} (x-\alpha_j) \cdot \prod_{\varepsilon=0}^{t-1} \left(y \prod_{j \in \Lambda_\varepsilon} (x-\alpha_j) - T_\varepsilon(x) \right), \end{aligned} \tag{47}$$