Low-Complexity Koetter-Vardy Decoding of Reed-Solomon Codes Using Module Minimization

Jiongyue Xing †, Li Chen ‡, Martin Bossert §
† School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou, China
‡ School of Electronics and Communication Engineering, Sun Yat-sen University, Guangzhou, China
§ Institute of Communications Engineering, Ulm University, Ulm, Germany
Email: xingjyue@mail2.sysu.edu.cn, chenli55@mail.sysu.edu.cn, martin.bossert@uni-ulm.de

***Abstract*—The Koetter-Vardy (KV) algorithm achieves advanced decoding performance for Reed-Solomon (RS) codes but with a high computational cost. This paper studies the low-complexity KV decoding that utilizes the module minimization (MM) interpolation technique, namely the KV-MM algorithm. A module contains bivariate polynomials that interpolate all the prescribed points with their multiplicity. Presenting the module basis as a matrix over univariate polynomials, row operation further reduces it into the Gröbner basis, delivering the interpolated polynomial. We will also introduce the re-encoding transformed KV-MM algorithm by giving an explicit construction for the module basis. This research shows MM interpolation yields a remarkably lower complexity for KV decoding than the conventional Koetter's interpolation, especially for high rate codes. This is also true when the re-encoding transform is applied. This finding is a rectification of some earlier results.**

***Index Terms*—Complexity reduction, Koetter-Vardy algorithm, module minimization, Reed-Solomon codes**

I. Introduction

Reed-Solomon (RS) codes are widely used in data communication systems and storage devices. In practice, the Berlekamp-Massey (BM) algorithm [1] is used for decoding. However, its error-correction capability is limited by half of the code's minimum Hamming distance. In late 90s, Guruswami and Sudan introduced the algebraic decoding algorithm, the so-called GS algorithm, which corrects errors beyond the above limit [2]. It has two steps, interpolation and root-finding. Koetter and Vardy [3] further introduced the algebraic soft decoding, the so-called KV algorithm. It yields remarkable performance gains over the BM and the GS algorithms with a polynomial-time complexity.

However, the algebraic decoding complexity is still orders of magnitude higher than the BM algorithm. This is due to the interpolation that is often realized by Koetter's iterative polynomial construction approach [4]. There exist several approaches to facilitate the interpolation, e.g., the re-encoding transform [5] and the progressive decoding [6]. It has been reported that the interpolation problem can also be solved from the perspective of Gröbner basis of module [7]. It formulates a basis of module which contains bivariate polynomials that interpolate all the prescribed points. The basis will then be reduced into the Gröbner basis which contains the interpolated polynomial. This interpolation technique is called the module minimization (MM) which refers to the basis reduction process. It can be further facilitated by several recent techniques [8] [9]. So far, performing the KV decoding using the MM technique has been sparsely reported in [7] [10], with its re-encoding transformed variant in [11] [12].

However, this MM based KV decoding still demands a more comprehensive study. On one hand, an explicit module basis construction is still needed, especially for the re-encoding transformed KV-MM algorithm. On the other hand, the complexity reduction effect brought by the MM interpolation and the re-encoding transform remains unknown for practical codes. Therefore, this paper provides a thorough research of the KV-MM algorithm and its re-encoding transformed variant. We will give an explicit module basis construction for both the KV-MM algorithm and its re-encoding transformed variant. To facilitate the understanding, the module basis construction will be illustrated by work examples. A simpler proof of the module generators is given. We will show that re-encoding transform reduces the MM complexity by reducing the degree of module generators. It should be pointed out that the earlier work [11] showed when only considering the finite field multiplication, the re-encoding transformed KV-MM algorithm yields a higher complexity than the case using Koetter's interpolation. However, our research shows by counting both finite field addition and multiplication, the MM interpolation can significantly reduce the decoding complexity over Koetter's interpolation, despite whether the re-encoding transform is applied. Especially for high rate codes, a complexity reduction of at least an order of magnitude can be achieved. To the best of our knowledge, this has never been reported in literature.

II. Prerequisite Knowledge

This section introduces some prerequisite knowledge for the paper, including RS codes and the KV decoding.

A. RS Codes

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

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

978-1-5386-8088-9/19/$31.00 ©2019 IEEE

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

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

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

B. The KV Decoding

Assume codeword $\underline{c} = (c_0, c_1, \ldots, c_{n-1})$ is transmitted through a memoryless channel and $\underline{\mathsf{r}} = (\mathsf{r}_0, \mathsf{r}_1, \ldots, \mathsf{r}_{n-1}) \in \mathbb{R}^n$ is the received vector. A $q \times n$ reliability matrix $\mathbf{\Pi}$ can be obtained based on $\underline{\mathsf{r}}$. Its entry $\pi_{ij} = \Pr[c_j = \sigma_i \mid \mathsf{r}_j]$ is the symbol wise *a posteriori* probability (APP) [1]. Matrix $\mathbf{\Pi}$ will be transformed into a multiplicity matrix $\mathbf{M}$ of the same size [3]. For $\mathbf{M}$, its entry m_{ij} is the interpolation multiplicity for point (α_j, σ_i). Interpolation determines the minimum polynomial $Q(x, y)$ that interpolates all prescribed points with their multiplicity. Let $i_j = \text{index}\{\sigma_i \mid \sigma_i = c_j\}$, the codeword score is defined as $S_{\mathbf{M}}(\underline{c}) = \sum_{j=0}^{n-1} m_{i_j j}$. Given a polynomial $Q(x, y) = \sum_{a,b} Q_{ab} x^a y^b \in \mathbb{F}_q[x, y]$, its monomials $x^a y^b$ can be organized under the (μ, ν)-revlex order [2]. Let $x^{a'} y^{b'}$ denote the leading monomial of Q where $Q_{a'b'} \neq 0$, 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 whose leading monomials are $x^{a_1'} y^{b_1'}$ and $x^{a_2'} y^{b_2'}$, respectively, it is claimed $Q_1 < Q_2$ if $x^{a_1'} y^{b_1'} < x^{a_2'} y^{b_2'}$. What follows is a sufficient condition for a successful KV decoding.

Theorem 1 [3]. For an (n, k) RS code, let $Q \in \mathbb{F}_q[x, y]$ denote an interpolated polynomial constructed based on $\mathbf{M}$. If $S_{\mathbf{M}}(\underline{c}) > \deg_{1,k-1} Q(x, y)$, $Q(x, f(x)) = 0$.

Interpolation finds the above polynomial Q with the minimum $(1, k-1)$-weighted degree. This is often realized by Koetter's algorithm [4] which is an iterative polynomial construction process. Root-finding further determines y-roots of Q which may contain the intended message $f(x)$ [13]. Hence, the maximum decoding output list size (OLS) would be $\deg_y Q$. A larger y-degree yields a stronger error-correction capability, but it also implies a higher complexity in computing Q. In this paper, we let $l = \deg_y Q$ to be the decoding parameter.

III. The KV-MM Algorithm

This section introduces the KV-MM algorithm, where the MM interpolation consists of module formulation and minimization. We first introduce the module for the KV decoding.

A. Module and the Gröbner Basis

For KV decoding with a maximum decoding OLS of l, a module $\mathcal{M}_l$ is needed. It is defined as follows.

Definition I. Module $\mathcal{M}_l$ is the space of all polynomials over $\mathbb{F}_q[x, y]$ that interpolate all points (α_j, σ_i) with a multiplicity of m_{ij} $(m_{ij} \neq 0)$. Their maximum y-degree is l.

[1]It is assumed that $\Pr[c_j = \sigma_i] = \frac{1}{q}, \forall (i, j)$.

[2]The (μ, ν)-weighted degree of $x^a y^b$ is $\deg_{\mu,\nu} x^a y^b = \mu a + \nu b$. Given two monomials $x^{a_1} y^{b_1}$ 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 $b_1 < b_2$.

$\mathcal{M}_l$ can be formulated by $l+1$ generators $P_t(x, y)$ where $t = 0, 1, \ldots, l$. They define the basis of the module. This basis $\mathcal{B}_l$ can be presented as a matrix over $\mathbb{F}_q[x]$. For this presentation, vectors over $\mathbb{F}_q[x]$ need to be defined.

Definition II. Let $\underline{\xi} = (\xi_\tau(x), \tau = 0, 1, \ldots)$ 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

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

Since $\xi_\tau(x) = \xi_\tau^{(0)} + \xi_\tau^{(1)} x + \cdots + \xi_\tau^{(\deg \xi_\tau(x))} x^{\deg \xi_\tau(x)}$, the leading term (LT) of $\xi_\tau(x)$ is

$$\text{LT}(\xi_\tau(x)) = \xi_\tau^{(\deg \xi_\tau(x))} x^{\deg \xi_\tau(x)}. \tag{5}$$

Given a matrix $\mathcal{V}$ over $\mathbb{F}_q[x]$, we denote its row-t as $\mathcal{V}|_t$ and its entry of row-t column-τ as $\mathcal{V}|_t^{(\tau)}$. For a module generator $P_t(x, y) = \sum_{\tau \leq l} P_t^{(\tau)}(x) y^\tau$ where $P_t^{(\tau)}(x) \in \mathbb{F}_q[x]$, it can be presented as a vector over $\mathbb{F}_q[x]$, i.e., $(P_t^{(0)}(x), P_t^{(1)}(x), \ldots, P_t^{(l)}(x))$. Therefore, basis $\mathcal{B}_l$ can be presented as a matrix over $\mathbb{F}_q[x]$ by letting $\mathcal{B}_l|_t^{(\tau)} = P_t^{(\tau)}(x), \forall (t, \tau)$. Note that $\mathcal{B}_l$ is a square matrix, which will become clear in Section III.B.

Definition III [14]. Given a square matrix $\mathcal{V}$ over $\mathbb{F}_q[x]$, if it exhibits $\text{LP}(\mathcal{V}|_t) \neq \text{LP}(\mathcal{V}|_{t'})$ for any two rows $\mathcal{V}|_t$ and $\mathcal{V}|_{t'}$, it is in the *weak Popov form*.

Given basis $\mathcal{B}_l$, it will be transformed into the Gröbner basis. The following Proposition gives a simple criterion for the Gröbner basis of $\mathcal{M}_l$.

Proposition 2 [7]. Assume that $\{g_t \in \mathbb{F}_q[x, y], 0 \leq t \leq l\}$ generates module $\mathcal{M}_l$. Under the (μ, ν)-revlex order, if y-degree of leading monomial of each polynomial g_t is different, $\{g_t \in \mathbb{F}_q[x, y], 0 \leq t \leq l\}$ is a Gröbner basis of $\mathcal{M}_l$.

After constructing basis $\mathcal{B}_l$, the MS algorithm [14] will reduce it into the Gröbner basis that is defined under $(1, k-1)$-revlex order. Before that, the mapping of

$$\mathcal{A}_l = \mathcal{B}_l \cdot \text{diag}(1, x^{k-1}, \ldots, x^{l(k-1)}) \tag{6}$$

will be performed. It enables $\deg \mathcal{A}_l|_t = \deg_{1,k-1} P_t(x, y)$. The MS algorithm then reduces $\mathcal{A}_l$ into the weak Popov form $\mathcal{A}_l'$ as follows. Find two rows $\mathcal{A}_l|_t$ and $\mathcal{A}_l|_{t'}$ such that $\deg \mathcal{A}_l|_t \leq \deg \mathcal{A}_l|_{t'}$ and $\text{LP}(\mathcal{A}_l|_t) = \text{LP}(\mathcal{A}_l|_{t'})$, and perform

$$\mathcal{A}_l|_{t'} = \mathcal{A}_l|_{t'} - \frac{\text{LT}(\mathcal{A}_l|_{t'}^{(\text{LP}(\mathcal{A}_l|_{t'}))})}{\text{LT}(\mathcal{A}_l|_t^{(\text{LP}(\mathcal{A}_l|_t))})} \cdot \mathcal{A}_l|_t. \tag{7}$$

Iterate this row operation until the weak Popov form $\mathcal{A}_l'$ is reached. Afterwards, demap $\mathcal{A}_l'$ as

$$\mathcal{B}_l' = \mathcal{A}_l' \cdot \text{diag}(1, x^{-(k-1)}, \ldots, x^{-l(k-1)}). \tag{8}$$

Now, $\mathcal{B}_l'$ becomes a Gröbner basis. Let $P_t'(x, y)$ be the polynomial that is retrieved from $\mathcal{B}_l'|_t$ by $P_t'^{(\tau)}(x) = \mathcal{B}_l'|_t^{(\tau)}$. Note that $\deg_{1,k-1} P_t'(x, y) = \deg \mathcal{A}_l'|_t = \deg \mathcal{A}_l'|_t^{(\text{LP}(\mathcal{A}_l'|_t))} = \deg P_t'^{(\text{LP}(\mathcal{A}_l'|_t))}(x) + (k-1) \cdot \text{LP}(\mathcal{A}_l'|_t)$. When $\mathcal{A}_l'$ is in the weak Popov form, y-degree of each polynomial's leading monomial,

$$\mathbf{M} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 4 & 0 & 0 & 2 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(a)

L_0	L_1	L_2	L_3	L_4	L_5	L_6
(α_0,σ_7)	(α_1,σ_3)	(α_2,σ_6)	(α_3,σ_1)	(α_4,σ_2)	(α_5,σ_4)	(α_6,σ_2)
(α_0,σ_7)	(α_1,σ_6)	(α_2,σ_6)	(α_3,σ_1)	(α_4,σ_2)	(α_5,σ_4)	(α_6,σ_3)
(α_0,σ_7)	(α_1,σ_6)	(α_2,σ_6)	(α_3,σ_1)	(α_4,σ_2)	(α_5,σ_6)	(α_6,σ_3)
(α_0,σ_7)	(α_1,σ_6)	(α_2,σ_6)	(α_3,σ_1)	(α_4,σ_4)	(α_5,σ_6)	

(b)

L'_0	L'_1	L'_2	L'_3	L'_4	L'_5	L'_6
(α_0,σ_7)	(α_1,σ_6)	(α_2,σ_6)	(α_3,σ_1)	(α_4,σ_2)	(α_5,σ_4)	(α_6,σ_3)
(α_0,σ_7)	(α_1,σ_6)	(α_2,σ_6)	(α_3,σ_1)	(α_4,σ_2)	(α_5,σ_6)	(α_6,σ_3)
(α_0,σ_7)	(α_1,σ_6)	(α_2,σ_6)	(α_3,σ_1)	(α_4,σ_2)	(α_5,σ_4)	(α_6,σ_2)
(α_0,σ_7)	(α_1,σ_3)	(α_2,σ_6)	(α_3,σ_1)	(α_4,σ_4)	(α_5,σ_6)	

(c)

L'_0	L'_1	L'_2	L'_3	L'_4	L'_5	L'_6
$(\alpha_0,0)$	$(\alpha_1,0)$	$(\alpha_2,0)$	$(\alpha_3,0)$	$(\alpha_4,0)$	(α_5,σ_7)	(α_6,σ_5)
$(\alpha_0,0)$	$(\alpha_1,0)$	$(\alpha_2,0)$	$(\alpha_3,0)$	$(\alpha_4,0)$	(α_5,σ_5)	(α_6,σ_5)
$(\alpha_0,0)$	$(\alpha_1,0)$	$(\alpha_2,0)$	$(\alpha_3,0)$	$(\alpha_4,0)$	(α_5,σ_7)	(α_6,σ_4)
$(\alpha_0,0)$	(α_1,σ_5)	$(\alpha_2,0)$	$(\alpha_3,0)$	(α_4,σ_6)	(α_5,σ_5)	

(d)

Fig. 1. (a) Multiplicity matrix; (b) Enumeration lists; (c) Balanced lists; (d) Transformed lists.

TABLE I
GENERATORS OF $\mathcal{M}_l$

$$P_0(x,y) = (x-\alpha_0)^4(x-\alpha_1)^3(x-\alpha_2)^4(x-\alpha_3)^4(x-\alpha_4)^3(x-\alpha_5)^2(x-\alpha_6)^2$$
$$P_1(x,y) = (x-\alpha_0)^3(x-\alpha_1)^2(x-\alpha_2)^3(x-\alpha_3)^3(x-\alpha_4)^2(x-\alpha_5)^2(x-\alpha_6)(y-F_0(x))$$
$$P_2(x,y) = (x-\alpha_0)^2(x-\alpha_1)(x-\alpha_2)^2(x-\alpha_3)^2(x-\alpha_4)(x-\alpha_5)(x-\alpha_6)(y-F_0(x))(y-F_1(x))$$
$$P_3(x,y) = (x-\alpha_0)(x-\alpha_1)(x-\alpha_2)(x-\alpha_3)(x-\alpha_4)(x-\alpha_5)(y-F_0(x))(y-F_1(x))(y-F_2(x))$$
$$P_4(x,y) = (y-F_0(x))(y-F_1(x))(y-F_2(x))(y-F_3(x))$$

$$F_0(x) = \sigma_7\Phi_0(x) + \sigma_6\Phi_1(x) + \sigma_6\Phi_2(x) + \sigma_1\Phi_3(x) + \sigma_2\Phi_4(x) + \sigma_4\Phi_5(x) + \sigma_3\Phi_6(x)$$
$$F_1(x) = \sigma_7\Phi_0(x) + \sigma_6\Phi_1(x) + \sigma_6\Phi_2(x) + \sigma_1\Phi_3(x) + \sigma_2\Phi_4(x) + \sigma_6\Phi_5(x) + \sigma_3\Phi_6(x)$$
$$F_2(x) = \sigma_7\Phi_0(x) + \sigma_6\Phi_1(x) + \sigma_6\Phi_2(x) + \sigma_1\Phi_3(x) + \sigma_2\Phi_4(x) + \sigma_4\Phi_5(x) + \sigma_2\Phi_6(x)$$
$$F_3(x) = \sigma_7\Phi_0(x) + \sigma_3\Phi_1(x) + \sigma_6\Phi_2(x) + \sigma_1\Phi_3(x) + \sigma_4\Phi_4(x) + \sigma_6\Phi_5(x)$$

i.e., LP($\mathcal{A}'_l|_t$), is different. Based on Proposition 2, $\mathcal{B}'_l$ is a Gröbner basis.

There exist several fast basis reduction approaches [8]–[10]. However, our research has shown for practical codes, e.g., the (255, 239) RS code, the MS algorithm requires less finite field arithmetic operations than the Alekhnovich algorithm [10].

B. Module Formulation and Minimization

Given matrix $\mathbf{M}$, we first define

$$\mathsf{m}_j = \sum_{i=0}^{q-1} m_{ij} \tag{9}$$

and $\mathsf{m} = \max\{\mathsf{m}_j, \forall j\}$. The $\mathbf{\Pi} \rightarrow \mathbf{M}$ transform terminates when $\mathsf{m} = l$. To formulate $\mathcal{M}_l$, the following point enumeration is needed. Let L_j denote an enumeration list that is drawn from column j of $\mathbf{M}$ as

$$L_j = [\underbrace{(\alpha_j,\sigma_i),\ldots,(\alpha_j,\sigma_i)}_{m_{ij}}, \forall i \text{ and } m_{ij} \neq 0]. \tag{10}$$

Note that $|L_j| = \mathsf{m}_j$. Its balanced list L'_j is further created as follows. Initialize $L'_j = \emptyset$. Move one of the most frequent elements from L_j to L'_j. Repeat this process m_j times until $L_j = \emptyset$. We denote the balanced list as

$$L'_j = [(\alpha_j, y_j^{(0)}), (\alpha_j, y_j^{(1)}), \ldots, (\alpha_j, y_j^{(\mathsf{m}_j-1)})], \tag{11}$$

where $y_j^{(0)}, y_j^{(1)}, \ldots, y_j^{(\mathsf{m}_j-1)} \in \mathbb{F}_q$ and they may not be distinct. Further let

$$\mathsf{m}_j(t) = \max\{\text{multi.}((\alpha_j, y_j^{(\varepsilon)})) \mid \varepsilon = t, t+1, \ldots, \mathsf{m}_j - 1\}. \tag{12}$$

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

The following example illustrates the above definitions.

Example 1. In decoding a (7, 5) RS code, the multiplicity matrix $\mathbf{M}$ is given as in 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. When $t = 0$, $\mathsf{m}_0(0) = 4$, $\mathsf{m}_1(0) = 3$, $\mathsf{m}_2(0) = 4$, $\mathsf{m}_3(0) = 4$, $\mathsf{m}_4(0) = 3$, $\mathsf{m}_5(0) = 2$ and $\mathsf{m}_6(0) = 2$. When $t = 1$, $\mathsf{m}_0(1) = 3$, $\mathsf{m}_1(1) = 2$, $\mathsf{m}_2(1) = 3$, $\mathsf{m}_3(1) = 3$, $\mathsf{m}_4(1) = 2$, $\mathsf{m}_5(1) = 2$ and $\mathsf{m}_6(1) = 1$.

Now, module $\mathcal{M}_l$ can be formulated. Let

$$F_\varepsilon(x) = \sum_{j=0}^{n-1} y_j^{(\varepsilon)} \Phi_j(x), \tag{13}$$

where $\varepsilon = 0, 1, \ldots, l-1$ and $\Phi_j(x) = \prod_{j'=0, j' \neq j}^{n-1} \frac{x-\alpha_{j'}}{\alpha_j - \alpha_{j'}}$ is the Lagrange basis polynomial. It holds $\Phi_j(\alpha_j) = 1$ and $\Phi_j(\alpha_{j'}) = 0, \forall j' \neq j$. Hence, $F_\varepsilon(\alpha_j) = y_j^{(\varepsilon)}, \forall j$. Polynomial $y - F_\varepsilon(x)$ interpolates points $(\alpha_j, y_j^{(\varepsilon)}), \forall j$. Note that if $\mathsf{m}_j < l$, we assume $y_j^{(\varepsilon)} = 0$ for $\varepsilon \geq \mathsf{m}_j$. Consequently, $\mathcal{M}_l$ can be generated as an $\mathbb{F}_q[x]$-module by

$$P_t(x,y) = \prod_{j=0}^{n-1} (x-\alpha_j)^{\mathsf{m}_j(t)} \prod_{\varepsilon=0}^{t-1} (y - F_\varepsilon(x)), \tag{14}$$

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. Since $\deg_y P_t(x,y) \le l, \forall t$, recalling Definition I, $P_t(x,y) \in \mathcal{M}_l$.

The following example further illustrates the above mentioned module formulation.

Example 2. Based on the balanced lists of *Example 1*, Table I shows the module generators in decoding the RS code.

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

Proof: Let $\mathcal{Q}_t(x,y) = \sum_{\tau=0}^{t} \mathcal{Q}_t^{(\tau)}(x)y^\tau \in \mathcal{M}_l$ where $\deg_y \mathcal{Q}_t = t$, it satisfies $\prod_{j=0}^{n-1}(x-\alpha_j)^{\mathsf{m}_j(t)} | \mathcal{Q}_t^{(t)}(x)$ [10]. Assume $\mathcal{Q}(x,y) \in \mathcal{M}_l$ and let us write (14) as $P_t(x,y) = \sum_{\tau=0}^{t} P_t^{(\tau)}(x)y^\tau$. When $t = l$, $P_l^{(l)}(x) = 1$, there exists a polynomial $p_l(x) \in \mathbb{F}_q[x]$ that enables $\mathcal{Q}_{l-1}(x,y) = \mathcal{Q}(x,y) - p_l(x)P_l(x,y)$ such that $\deg_y \mathcal{Q}_{l-1} = l-1$. Note that if $\deg_y \mathcal{Q} < l$, $p_l(x) = 0$. Since $(\mathcal{Q}, P_l) \in \mathcal{M}_l$, $\mathcal{Q}_{l-1} \in \mathcal{M}_l$. When $t = l-1$, $P_{l-1}^{(l-1)}(x) = \prod_{j=0}^{n-1}(x-\alpha_j)^{\mathsf{m}_j(l-1)}$ and $\prod_{j=0}^{n-1}(x-\alpha_j)^{\mathsf{m}_j(l-1)} | \mathcal{Q}_{l-1}^{(l-1)}(x)$. Therefore, we can generate $\mathcal{Q}_{l-2}(x,y)$ by $\mathcal{Q}_{l-2}(x,y) = \mathcal{Q}_{l-1}(x,y) - p_{l-1}(x)P_{l-1}(x,y)$ such that $\deg_y \mathcal{Q}_{l-2} = l-2$. Following the above deduction until $t = 0$, we have $P_0^{(0)}(x) = \prod_{j=0}^{n-1}(x-\alpha_j)^{\mathsf{m}_j(0)}$ and $\prod_{j=0}^{n-1}(x-\alpha_j)^{\mathsf{m}_j(0)} | \mathcal{Q}_0^{(0)}(x)$. Hence, there exists $p_0(x)$ that enables $\mathcal{Q}_0(x,y) - p_0(x)P_0(x,y) = 0$. Therefore, if $\mathcal{Q} \in \mathcal{M}_l$, it can be written as an $\mathbb{F}_q[x]$-linear combination of $P_t(x,y)$. ■

Theorem 3 reveals that $P_t(x,y)$ of (14) forms a basis $\mathcal{B}_l$ of $\mathcal{M}_l$. Since $\deg_y P_t(x,y) \le l$, $\mathcal{B}_l$ can be presented as an $(l+1)\times(l+1)$ square matrix over $\mathbb{F}_q[x]$. Perform the mapping of (6) to yield $\mathcal{A}_l$. The MS algorithm reduces $\mathcal{A}_l$ into $\mathcal{A}'_l$. Demap it as (8) and $\mathcal{B}'_l$ is the Gröbner basis. Let $\mathcal{A}'_l|_{t^*}$ denote the minimum row of $\mathcal{A}'_l$, the interpolated polynomial $Q(x,y) = \sum_{\tau \le l} Q^{(\tau)}(x)y^\tau$ can be retrieved from $\mathcal{B}'_l|_{t^*}$ by

$$Q^{(\tau)}(x) = \mathcal{B}'_l|_{t^*}^{(\tau)}. \tag{15}$$

The root-finding further determines y-roots of Q, yielding the estimated message polynomial $\hat{f}(x)$.

The KV-MM algorithm is summarized as follows.

Algorithm 1 The KV-MM Algorithm

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

1: Create all balanced lists L'_j as in (11);
2: Formulate $\mathcal{B}_l$ by (14) and map it to $\mathcal{A}_l$ by (6);
3: Reduce $\mathcal{A}_l$ into $\mathcal{A}'_l$ and demap it to $\mathcal{B}'_l$ by (8);
4: Construct Q by (15);
5: Retrieve y-roots of Q to find $\hat{f}(x)$.

IV. The Re-encoding Transformed KV-MM

This section introduces the re-encoding transformed KV-MM algorithm that yields a further reduced complexity.

A. Re-encoding Transform

Re-encoding transform will result in reducing the x-degree of module generators. This leads to a smaller MS complexity. Let us sort $\mathsf{m}_0(0), \mathsf{m}_1(0), \ldots, \mathsf{m}_{n-1}(0)$ to obtain an index sequence $j_0, j_1, \ldots, j_{n-1}$, which indicates $\mathsf{m}_{j_0}(0) \ge \mathsf{m}_{j_1}(0) \ge \cdots \ge \mathsf{m}_{j_{n-1}}(0)$. 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 $(\alpha_j, y_j^{(0)})$ where $j \in \Upsilon$ are chosen to construct the re-encoding polynomial

$$H(x) = \sum_{j\in\Upsilon} y_j^{(0)} \prod_{j'\in\Upsilon, j'\ne j} \frac{x-\alpha_{j'}}{\alpha_j - \alpha_{j'}}. \tag{16}$$

Hence, $H(\alpha_j) = y_j^{(0)}, \forall j \in \Upsilon$. All interpolation points $(\alpha_j, y_j^{(\varepsilon)})$ in the balanced lists are transformed by

$$(\alpha_j, w_j^{(\varepsilon)}) = (\alpha_j, y_j^{(\varepsilon)} - H(\alpha_j)). \tag{17}$$

Note that for $j \in \Upsilon$, if $y_j^{(\varepsilon)} = y_j^{(0)}$, $w_j^{(\varepsilon)} = 0$.

B. Module Formulation and Minimization

The module will be formulated based on the transformed balanced lists. Let $\Lambda_\varepsilon = \{j \mid w_j^{(\varepsilon)} = 0, j \in \Upsilon\}$ and $\bar{\Lambda}_\varepsilon = \Upsilon \backslash \Lambda_\varepsilon$. With the transformed points $(\alpha_j, w_j^{(\varepsilon)})$, polynomials $F_\varepsilon(x)$ of (13) is redefined as

$$F_\varepsilon(x) = \sum_{j=0}^{n-1} w_j^{(\varepsilon)} \Phi_j(x). \tag{18}$$

Let

$$\phi(x) = \prod_{j\in\Upsilon} (x-\alpha_j)^{\mathsf{m}_j(0)} \tag{19}$$

and

$$\psi(x) = \prod_{j\in\Upsilon} (x-\alpha_j). \tag{20}$$

The following Theorem reveals the property of module generators (14) when the re-encoding transform is applied.

Theorem 4. Given matrix $\mathbf{M}$ and the transformed balanced lists, $\phi(x) | P_t(x, y\psi(x))$.

Proof: The proof is given in Appendix A of [15]. ■

Therefore, we can define the following bijective mapping

$$\begin{aligned} \varphi: \quad & \mathcal{M}_l \to \mathbb{F}_q[x,y] \\ & P_t(x,y) \mapsto \phi(x)^{-1} P_t(x, y\psi(x)), \end{aligned} \tag{21}$$

where φ is an isomorphism between $\mathcal{M}_l$ and $\varphi(\mathcal{M}_l)$. Polynomials of $\varphi(\mathcal{M}_l)$ have lower x-degree than those of $\mathcal{M}_l$. Since the MS basis reduction algorithm performs linear combination between its polynomials, the mapping of (21) will result in a simpler basis reduction process.

Further let

$$\tilde{w}_j^{(\varepsilon)} = \frac{w_j^{(\varepsilon)}}{\zeta_j}, \tag{22}$$

where $\zeta_j = \prod_{j'=0, j'\ne j}^{n-1}(\alpha_j - \alpha_{j'})$, we define

$$T_\varepsilon(x) = \sum_{j\in\bar{\Upsilon}\cup\bar{\Lambda}_\varepsilon} \tilde{w}_j^{(\varepsilon)} \prod_{j'\in\bar{\Upsilon}\cup\bar{\Lambda}_\varepsilon, j'\ne j} (x-\alpha_{j'}). \tag{23}$$

TABLE II
GENERATORS OF $\varphi(\mathcal{M}_l)$

$\tilde{P}_0(x,y) = (x-\alpha_5)^2(x-\alpha_6)^2$	$T_0(x) = \frac{\sigma_7}{\zeta_5}(x-\alpha_6) + \frac{\sigma_5}{\zeta_6}(x-\alpha_5)$
$\tilde{P}_1(x,y) = (x-\alpha_5)^2(x-\alpha_6)(y-T_0(x))$	$T_1(x) = \frac{\sigma_5}{\zeta_5}(x-\alpha_6) + \frac{\sigma_5}{\zeta_6}(x-\alpha_5)$
$\tilde{P}_2(x,y) = (x-\alpha_5)(x-\alpha_6)(y-T_0(x))(y-T_1(x))$	$T_2(x) = \frac{\sigma_7}{\zeta_5}(x-\alpha_6) + \frac{\sigma_4}{\zeta_6}(x-\alpha_5)$
$\tilde{P}_3(x,y) = (x-\alpha_1)(x-\alpha_4)(x-\alpha_5)(y-T_0(x))(y-T_1(x))(y-T_2(x))$	$T_3(x) = \frac{\sigma_5}{\zeta_1}(x-\alpha_4)(x-\alpha_5)(x-\alpha_6) + \frac{\sigma_6}{\zeta_4}(x-\alpha_1)(x-\alpha_5)$
$\tilde{P}_4(x,y) = (y-T_0(x))(y-T_1(x))(y-T_2(x))(y(x-\alpha_1)(x-\alpha_4) - T_3(x))$	$(x-\alpha_6) + \frac{\sigma_5}{\zeta_5}(x-\alpha_1)(x-\alpha_4)(x-\alpha_6)$

The proof of Theorem 4 shows that generators of $\varphi(\mathcal{M}_l)$ can be constructed by

$$\tilde{P}_t(x,y) = \prod_{j\in\bar{\Upsilon}}(x-\alpha_j)^{\mathsf{m}_j(t)} \cdot \prod_{j\in\bar{\Lambda}_t}(x-\alpha_j) \cdot \prod_{\varepsilon=0}^{t-1}\Big(y\prod_{j\in\bar{\Lambda}_\varepsilon}(x-\alpha_j) - T_\varepsilon(x)\Big), \quad (24)$$

where $t = 0, 1, \ldots, l$. Note that $\bar{\Lambda}_l = \emptyset$. Based on Theorem 3 and (21), we know (24) also forms a basis $\tilde{\mathcal{B}}_l$ of $\varphi(\mathcal{M}_l)$. Again, it can be presented as a square matrix over $\mathbb{F}_q[x]$.

The following example further illustrates the above module formulation.

Example 3. We continue from *Example 1*. By sorting $\mathsf{m}_0(0), \mathsf{m}_1(0), \ldots, \mathsf{m}_6(0)$, we have $\Upsilon = \{0,1,2,3,4\}$ and $\bar{\Upsilon} = \{5,6\}$. Hence, (α_0, σ_7), (α_1, σ_6), (α_2, σ_6), (α_3, σ_1) and (α_4, σ_2) are chosen to generate the re-encoding polynomial $H(x) = \sigma_1 + \sigma_5 x^2 + \sigma_3 x^{3}$ [3]. The transformed balanced lists $L'_0 \sim L'_6$ are shown in Fig. 1 (d). We define $\phi(x) = (x-\alpha_0)^4(x-\alpha_1)^3(x-\alpha_2)^4(x-\alpha_3)^4(x-\alpha_4)^3$ and $\psi(x) = (x-\alpha_0)(x-\alpha_1)(x-\alpha_2)(x-\alpha_3)(x-\alpha_4)$. Generators of $\varphi(\mathcal{M}_l)$ are shown in Table II. Compared with Table I, module generators of $\varphi(\mathcal{M}_l)$ have smaller x-degree than those of $\mathcal{M}_l$.

After the re-encoding transform, polynomials are ordered under the $(1,-1)$-revlex order [5]. However, performing $\mathcal{A}_l = \tilde{\mathcal{B}}_l \cdot \mathrm{diag}(1, x^{-1}, \ldots, x^{-l})$ causes some of the basis entries leaving $\mathbb{F}_q[x]$. Alternatively, $\mathcal{A}_l$ will be generated by

$$\mathcal{A}_l = \tilde{\mathcal{B}}_l \cdot \mathrm{diag}(x^l, x^{l-1}, \ldots, 1), \quad (25)$$

such that $\deg \mathcal{A}_l|_t = \deg_{1,-1} \tilde{P}_t(x,y) + l$. The MS algorithm further reduces $\mathcal{A}_l$ into $\mathcal{A}'_l$. Demap it as

$$\tilde{\mathcal{B}}'_l = \mathcal{A}'_l \cdot \mathrm{diag}(x^{-l}, x^{-(l-1)}, \ldots, 1). \quad (26)$$

Again, if $\mathcal{A}'_l|_{t^*}$ is the minimum row, polynomial $\tilde{Q}(x,y) = \sum_{\tau\le l}\tilde{Q}^{(\tau)}(x)y^\tau$ can be retrieved from $\tilde{\mathcal{B}}'_l|_{t^*}$ by $\tilde{Q}^{(\tau)}(x) = \tilde{\mathcal{B}}'_l|^{(\tau)}_{t^*}$. Based on (21), the interpolated polynomial Q will be restored by

$$Q(x,y) = \phi(x)\tilde{Q}\Big(x, \frac{y}{\psi(x)}\Big). \quad (27)$$

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

[3]It is assumed $\mathbb{F}_8$ is defined by the primitive polynomial $\delta^3+\delta+1$, where δ is the primitive element. Moreover, $\mathbb{F}_8 = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_7\} = \{0, 1, \delta, \delta^3, \delta^2, \delta^6, \delta^4, \delta^5\}$.

The re-encoding transformed KV-MM algorithm is summarized as follows.

Algorithm 2 The Re-encoding Transformed KV-MM

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

1: Generate all balanced lists L'_j as in (11);
2: Sort $\mathsf{m}_0(0), \mathsf{m}_1(0), \ldots, \mathsf{m}_{n-1}(0)$ and define Υ;
3: Transform all interpolation points as in (17);
4: Formulate $\tilde{\mathcal{B}}_l$ by (24) and map it to $\mathcal{A}_l$ by (25);
5: Reduce $\mathcal{A}_l$ into $\mathcal{A}'_l$ and demap it to $\tilde{\mathcal{B}}'_l$ by (26);
6: Determine Q as in (27);
7: Retrieve y-roots of Q to further determine $\hat{f}(x)$.

V. DECODING PERFORMANCE AND COMPLEXITY

This section provides the decoding frame error rate (FER) and complexity performances of the KV-MM algorithms. They are compared with the KV algorithms that employ Koetter's interpolation, which are denoted as the KV-Koetter algorithms. In this paper, complexity is measured as the number of finite field arithmetic operations (multiplication and addition) in decoding a codeword, including the root-finding and the re-encoding transform. Our numerical results are obtained over the additive white Gaussian noise (AWGN) channel using BPSK modulation.

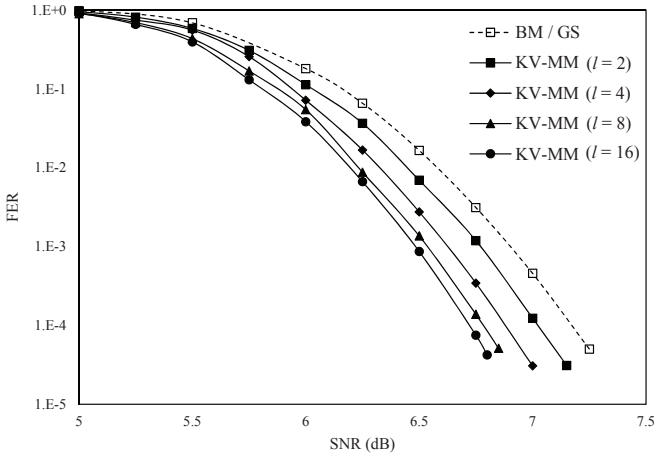

Fig. 2. Performance of the (255, 239) RS code.

Fig. 2 shows the KV-MM performance of the popular (255, 239) RS code. It can be seen that the KV-MM algorithm

TABLE III
KV DECODING COMPLEXITY OF SEVERAL RS CODES

		$l=4$				$l=8$			
		(63, 31)	(63, 55)	(255, 144)	(255, 239)	(63, 31)	(63, 55)	(255, 144)	(255, 239)
without re-encoding	KV-MM	1.82×10^6	1.15×10^6	3.13×10^7	2.75×10^7	3.01×10^7	1.56×10^7	5.04×10^8	3.33×10^8
	KV-Koetter	1.59×10^7	1.92×10^7	8.58×10^8	1.10×10^9	3.50×10^8	4.06×10^8	1.99×10^{10}	2.47×10^{10}
with re-encoding	KV-MM	1.48×10^6	5.87×10^5	2.02×10^7	1.63×10^7	1.11×10^7	4.10×10^6	2.54×10^8	1.27×10^8
	KV-Koetter	6.16×10^6	3.92×10^6	1.69×10^8	1.06×10^8	1.10×10^8	4.31×10^7	2.91×10^9	3.24×10^8

outperforms the BM and the GS algorithms, where the GS algorithm decodes with an interpolation multiplicity of one. The performance gain can be improved by increasing the decoding parameter l, i.e., $\deg_y Q$. However, this will be at the cost of decoding complexity.

The KV-MM complexity is dominated by the interpolation which contains module formulation and minimization. Our analysis [15] shows that when l is sufficiently large, the module minimization dominates the complexity. Therefore, when l is sufficiently large, the interpolation complexity is characterized by the MS algorithm. This will be determined by $\deg \mathcal{A}_l|_t^{(\tau)}$ and the number of row operations for reducing $\mathcal{A}_l$ into $\mathcal{A}_l'$. In [15], we analyze that without the re-encoding transform, $\deg \mathcal{A}_l|_t^{(\tau)} \leq nl$, and with the re-encoding transform, $\deg \mathcal{A}_l|_t^{(\tau)} \leq (n-k+1)l$. In both cases, the number of row operations is upper bounded by $\frac{1}{2}(n-k)(l+1)^3$. Since $\mathcal{A}_l$ is an $(l+1)\times(l+1)$ matrix over $\mathbb{F}_q[x]$, without the re-encoding transform, the MS complexity is $\frac{1}{2}n(n-k)(l+1)^5$. With the re-encoding transform, it is reduced to $\frac{1}{2}(n-k)^2(l+1)^5$. Therefore, re-encoding transform yields a complexity reduction factor of $\frac{k}{n}$. More importantly, the analysis shows both the MM interpolation and the re-encoding transform are more effective in yielding a low complexity for high rate codes, which is of practical interest.

Table III shows our numerical results of KV decoding of several RS codes. Without the re-encoding transform, the KV-MM algorithm is less complex than the KV-Koetter algorithm by at least an order of magnitude. High rate codes exhibit a smaller KV-MM complexity, which is opposite to the KV-Koetter algorithm. The re-encoding transform further reduces the KV-MM and the KV-Koetter complexity. In this case, decoding high rate codes would be less complex in using both interpolation techniques. This is because the complexity reduction essentially comes from k re-encoding points. The complexity reduction factor of $\frac{k}{n}$ holds for both the KV-MM and the KV-Koetter algorithms. Moreover, the KV-MM algorithm is still less complex than the KV-Koetter algorithm. This is a rectification of the earlier results of [11] which only considered finite field multiplication.

VI. CONCLUSIONS

This paper has introduced the low-complexity KV-MM algorithms for RS codes. Explicit constructions for module basis have been presented and illustrated by work examples. Our analysis and numerical results have verified their low-complexity feature in comparison with the cases using Koetter's interpolation. We have also 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 practical applications in which high rate codes are favored.

ACKNOWLEDGEMENT

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

REFERENCES

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

[2] 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.

[3] 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.

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

[5] R. Koetter and A. Vardy, "A complexity reducing transformation in algebraic list decoding of Reed-Solomon codes," in *Proc. the IEEE Inform. Theory Workshop (ITW)*, Paris, France, Apr. 2003, pp. 10–13.

[6] 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.

[7] 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)*, Seattle, USA, Jul. 2006, pp. 2032–2036.

[8] M. Chowdhury, C. Jeannerod, V. Neiger, É. Schost, and G. Villard, "Faster algorithms for multivariate interpolation with multiplicities and simultaneous polynomial approximations," *IEEE Trans. Inform. Theory*, vol. 61, no. 5, pp. 2370–2387, May 2015.

[9] C. Jeannerod, V. Neiger, É. Schost, and G. Villard, "Computing minimal interpolation bases," *J. Symb. Comput.*, vol. 83, pp. 272–314, 2017.

[10] 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.

[11] 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)*, Nice, France, Jun. 2007, pp. 1986–1990.

[12] X. Zhang and J. Zhu, "Low-complexity interpolation architecture for soft-decision Reed-Solomon decoding," in *Proc. the IEEE Int. Symp. Cir. Syst. (ISCAS)*, New Orleans, USA, May 2007, pp. 1413–1416.

[13] 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.

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

[15] J. Xing, L. Chen, and M. Bossert, "Module minimization based low-complexity soft decoding of Reed-Solomon codes," submitted to *IET Commun.*, 2019.