Low-Complexity Chase Decoding of Reed-Solomon Codes Using Module

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

***Abstract*— The interpolation based algebraic soft decoding yields a high decoding performance for Reed-Solomon (RS) codes with a polynomial-time complexity. Its computationally expensive interpolation can be facilitated using the module structure. The desired Gröbner basis can be achieved by reducing the basis of a module. This paper proposes the low-complexity Chase (LCC) decoding algorithm using this module basis reduction (BR) interpolation technique, namely the LCC-BR algorithm. By identifying η unreliable symbols, 2^{η} decoding test-vectors will be formulated. The LCC-BR algorithm first constructs a common basis which will be shared by the decoding of all test-vectors. This eliminates the redundant computation in decoding each test-vector, resulting in a lower decoding complexity and latency. This paper further proposes the progressive LCC-BR algorithm that decodes the test-vectors sequentially and terminates once the maximum-likelihood decision decoding outcome is reached. Exploiting the difference between the adjacent test-vectors, this progressive decoding is realized without any additional memory cost. Complexity analysis shows that the LCC-BR algorithm yields a lower complexity and latency, especially for high rate codes, which will be validated by the numerical results.**

***Index Terms*— Basis reduction, low-complexity Chase decoding, progressive decoding, Reed-Solomon codes.**

I. Introduction

REED-SOLOMON (RS) codes are among the most popular error-correction codes in data communications and storage systems. Currently, the Berlekamp-Massey (BM) algorithm [1], [2] is employed in the practical systems due to its effectiveness. It is a syndrome based decoding that delivers at most one message candidate. Hence, it is referred as the unique decoding. Other unique decoding algorithms include the extended Euclidean algorithm [3] and the Welch-Berlekamp algorithm [4]. They have the efficient running time but with the error-correction capability limited by half of the code's minimum Hamming distance. Utilizing soft information, the generalized minimum-distance decoding (GMD) algorithm [5] declares unreliable symbols as erasures and performs the error-erasure decoding, while the Chase algorithm [6] modifies the unreliable symbols and constitutes multiple decoding events. They both improve the error-correction performance.

In late 90s, breakthrough in correcting errors beyond the half distance bound was made by Guruswami and Sudan [7]. They proposed an interpolation based algebraic hard-decision decoding algorithm with a polynomial-time complexity. Later, Kötter and Vardy [8] introduced the algebraic soft-decision decoding algorithm by converting the symbol reliability into the interpolation multiplicity. However, their complexity remains high due to the construction of the interpolation polynomial, which can be realized using Kötter's iterative polynomial construction algorithm [9]. Another interpolation approach is based on the concept of Gröbner basis of a module [10]. It first constructs a polynomial basis of a module which satisfies all interpolation constraints, and then reduces the basis into a Gröbner basis for finding the interpolation polynomial. This technique is called basis reduction (BR).[1] The construction of a module basis was introduced in [11]. In particular, Lee and O'Sullivan proposed the explicit basis construction for the modules of the Guruswami-Sudan (GS) and the Kötter-Vardy (KV) algorithms in [12] and [13], respectively. There exist several efficient approaches [14]–[16] to realize the basis reduction process. The interpolation step can be transformed to a specific instance of M-Padé approximation and tackled by the fast matrix computation [17]–[19]. Besides, the interpolation problem can also be reformulated into a system of linear equations and solved by the structured linear algebra [20]–[22], among which the algorithm of [22] exhibits the best complexity characteristics.

Several techniques have been proposed to reduce the algebraic decoding complexity, such as the re-encoding transform [23]–[26] and the progressive interpolation [27]–[29]. The former approach was applied in Kötter's interpolation [23], [24]

Manuscript received November 2, 2019; revised April 4, 2020 and June 28, 2020; accepted July 17, 2020. Date of publication July 27, 2020; date of current version October 16, 2020. 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. This article was presented in part at the IEEE International Symposium on Information Theory (ISIT), 2020. The associate editor coordinating the review of this article and approving it for publication was P. Trifonov. *(Corresponding author: Li Chen.)*

Jiongyue Xing and Li Chen are with the School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou 510006, China (e-mail: xingjyue@mail2.sysu.edu.cn; chenli55@mail.sysu.edu.cn).

Martin Bossert is with the Institute of Communications Engineering, Ulm University, 89081 Ulm, Germany (e-mail: martin.bossert@uni-ulm.de).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCOMM.2020.3011991

[1] In our earlier publications, we named this technique module minimization (MM). However, during the revision, a more subtle consideration for the decoding technique is reached, thanking to one of the reviewers. Note that what being reduced is the basis of a module. Hence, "basis reduction (BR)" will be a more appropriate name for this interpolation technique. But for the sake of consistency, we will continue to use the acronym MM when we refer to our earlier proposed algorithms. The audience should be aware that both BR and MM mean the same interpolation technique.

and the BR (or module minimization (MM) as we used to call) interpolation [25], [26] for the KV algorithm, respectively. The latter was introduced in [27]–[29] to adjust the decoding computation to the level of corruption of the received information, resulting in the decoding complexity being channel dependent. By identifying η unreliable received symbols, the low-complexity Chase (LCC) decoding algorithm [30] formulates 2^η test-vectors and yields a competent decoding performance. This formulation also enables Kötter's interpolation to be performed in a binary-tree expansion manner for all test-vectors, resulting in a low-complexity feature. By further arranging the test-vectors such that the adjacent test-vectors only differ by one symbol, the backward-forward LCC (BF-LCC) algorithm [31] offers a hardware friendly decoding mechanism. Ordering the 2^η test-vectors, the progressive LCC algorithm [32] decodes the one that is more likely to yield the intended message. It terminates once the intended message is decoded [33], yielding a lower complexity when the received information is less corruptive.

Recently, the MM based algebraic Chase decoding (ACD-MM) algorithm [34] has been proposed. In contrast to the LCC algorithm [30], it can perform parallel decoding for each test-vector, lowering the decoding latency. However, similarity among the test-vectors has not been fully utilized, resulting in redundant computation. Addressing this problem, this paper proposes the BR based LCC (LCC-BR) algorithm for realizing the low-complexity and low-latency decoding of RS codes. We will show that the BR interpolation can be partitioned into the common basis construction and the individual basis construction. The former is performed once and its outcome will be shared by the individual basis construction for decoding each test-vector, fully eliminating the redundant computation of decoding all test-vectors. As the channel condition improves, e.g., the signal-to-noise ratio (SNR) increases, the received soft information becomes less corruptive. Intuitively, fewer test-vectors need to be decoded in retrieving the intended message. This paper further proposes the progressive LCC-BR algorithm, which decodes the test-vectors in a sequential manner and terminates once a maximum-likelihood (ML) codeword is decoded. In comparison with the progressive LCC algorithm [32], the proposal eliminates the need of memorizing the intermediate decoding information. We will reveal the low-complexity and low-latency features of the LCC-BR algorithm. This analysis shows that the BR interpolation technique will be more effective for high rate codes. Numerical results will show that the LCC-BR algorithm yields a similar complexity as the LCC algorithm [30], but a much lower decoding latency due to its parallel decoding feature. Average complexity of the progressive LCC-BR algorithm is further characterized, unveiling its channel dependent feature. Simulation results will show the complexity and latency advantages of the proposed algorithms over the existing ones.

II. Background Knowledge

This section presents the prerequisites of the paper, including the RS encoding and the MM based GS algorithm.

A. RS Encoding

Let $\mathbb{F}_q = \{\sigma_0, \sigma_1, \ldots, \sigma_{q-1}\}$ denote the finite field of size q, and $\mathbb{F}_q[x]$ and $\mathbb{F}_q[x, y]$ denote the univariate and the bivariate polynomial rings defined over $\mathbb{F}_q$, respectively. For an (n, k) RS code with length $n = q - 1$ and dimension k, message symbols $f_0, f_1, \ldots, f_{k-1}$ constitute a message polynomial as

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

The corresponding codeword $\underline{c} = (c_0, c_1, \ldots, c_{n-1})$ can be generated by

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

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

B. The MM Based GS Algorithm

Let $\underline{\omega} = (\omega_0, \omega_1, \ldots, \omega_{n-1}) \in \mathbb{F}_q^n$ denote the received word. The Hamming distance between $\underline{c}$ and $\underline{\omega}$ is $d_{\mathrm{H}}(\underline{c}, \underline{\omega}) = |\{j \mid c_j \neq \omega_j, \forall j\}|$. Given a polynomial $Q(x, y) = \sum_{a,b} Q_{ab} x^a y^b \in \mathbb{F}_q[x, y]$ and a nonnegative integer pair (κ, ι), the (κ, ι)-Hasse derivative evaluation at one point (α_j, ω_j) is defined as [35]

$$D_{\kappa,\iota}^{(\alpha_j,\omega_j)}(Q(x, y)) = \sum_{a \geq \kappa, b \geq \iota} \binom{a}{\kappa} \binom{b}{\iota} Q_{ab} \alpha_j^{a-\kappa} \omega_j^{b-\iota}.$$

If $D_{\kappa,\iota}^{(\alpha_j,\omega_j)}(Q(x, y)) = 0, \forall \kappa + \iota < m$, $Q(x, y)$ interpolates the point (α_j, ω_j) with a multiplicity m. The GS algorithm [7] first constructs an interpolation polynomial $Q(x, y)$ that passes through the n points $(\alpha_0, \omega_0), (\alpha_1, \omega_1), \ldots, (\alpha_{n-1}, \omega_{n-1})$ with a multiplicity m. The message polynomial $f(x)$ can be recovered by finding its y-roots, i.e., $Q(x, f(x)) = 0$ [36]. Hence, the maximum decoding output list size is determined by its y-degree, i.e., $\deg_y Q$. Let $l = \deg_y Q$ denote the decoding parameter. Note that $m \leq l$.

Definition I: Given a bivariate monomial $x^a y^b$, its (μ, ν)-weighted degree is $\deg_{\mu,\nu} x^a y^b = \mu a + \nu b$. Consequently, the (μ, ν)-reverse lexicographic (revlex) order is defined as follows. 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 $b_1 < b_2$.

Definition II: Given $Q(x, y) = \sum_{a,b} Q_{ab} x^a y^b \in \mathbb{F}_q[x, y]$, if $x^{a'} y^{b'} (Q_{a'b'} \neq 0)$ is the leading monomial (LM), the (μ, ν)-weighted degree of Q is $\deg_{\mu,\nu} Q = \deg_{\mu,\nu} x^{a'} y^{b'}$. Given two polynomials Q_1 and Q_2 with $\mathrm{LM}_{\mu,\nu}(Q_1) = x^{a_1'} y^{b_1'}$ and $\mathrm{LM}_{\mu,\nu}(Q_2) = x^{a_2'} y^{b_2'}$, respectively, $Q_1 < Q_2$ if $\mathrm{LM}_{\mu,\nu}(Q_1) < \mathrm{LM}_{\mu,\nu}(Q_2)$.

Consequently, the GS decoding theorem can be introduced.

Theorem 1 [7]: 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 m. If $m(n - d_{\mathrm{H}}(\underline{c}, \underline{\omega})) > \deg_{1,k-1} Q(x, y)$, $Q(x, f(x)) = 0$.

The interpolation polynomial Q can be determined by the MM approach [12]. It first constructs a basis of a module, which will then be reduced into a Gröbner basis that contains Q. The following defines a module $\mathcal{M}$.

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

Note that $\mathcal{M}$ is a free $\mathbb{F}_q[x]$-module of rank l [12]. In order to generate $\mathcal{M}$, its explicit basis needs to be constructed. Firstly, two univariate polynomials, namely the module seeds, are introduced as

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

and

$$R(x) = \sum_{j=0}^{n-1}\omega_j T_j(x), \tag{2}$$

where

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

is the Lagrange basis polynomial. It satisfies $T_j(\alpha_j) = 1$ and $T_j(\alpha_{j'}) = 0, \forall j' \neq j$. As a result, $R(\alpha_j) = \omega_j, \forall j$. With a multiplicity m and a decoding output list size l, $\mathcal{M}$ can be generated as an $\mathbb{F}_q[x]$-module by the following $l+1$ polynomials [12]

$$P_t(x,y) = G(x)^{m-t}(y-R(x))^t, \quad \text{if } 0 \le t \le m, \tag{3}$$

$$P_t(x,y) = y^{t-m}(y-R(x))^m, \quad \text{if } m < t \le l. \tag{4}$$

Let $\mathcal{B} = \{P_0(x,y), P_1(x,y), \ldots, P_l(x,y)\}$ define a polynomial set. Since $P_t(\alpha_j, \omega_j) = 0, \forall (t,j)$ and the total exponent of $G(x)$ and $y - R(x)$ is at least m, each $P_t(x,y)$ interpolates the n points with a multiplicity m. Furthermore, $\deg_y P_t(x,y) = t \le l$. Therefore, $\mathcal{B}$ spans the module $\mathcal{M}$, resulting in a basis of $\mathcal{M}$.

Lemma 2 [12]: Given a polynomial $\mathcal{Q}(x,y) \in \mathcal{M}$, it can be presented as an $\mathbb{F}_q[x]$-linear combination of $P_t(x,y)$, i.e., $\mathcal{Q}(x,y) = \sum_{t=0}^{l} p_t(x) \cdot P_t(x,y)$, where $p_t(x) \in \mathbb{F}_q[x]$.

The basis $\mathcal{B}$ will be reduced into a Gröbner basis, in which the minimum candidate is chosen as the interpolation polynomial $Q(x,y)$. The following Lemma describes a simple criterion to validate a Gröbner basis of a module.

Lemma 3 [12]: Assume that $\mathcal{G} = \{g_t \in \mathbb{F}_q[x,y], 0 \le t \le l\}$ is a basis of $\mathcal{M}$. If $\deg_y \mathrm{LM}_{\mu,\nu}(g_t) \neq \deg_y \mathrm{LM}_{\mu,\nu}(g_{t'}), \forall t \neq t'$, $\mathcal{G}$ is a Gröbner basis of $\mathcal{M}$.

The basis reduction is to perform $\mathbb{F}_q[x]$-linear combinations on the polynomials $P_t(x,y)$ until the y-degree of all $\mathrm{LM}_{\mu,\nu}(P_t)$ are different. There exist several approaches for the basis reduction, with an asymptotic complexity of $O(ml^4n^2)$ [14], $O(ml^4 n \log^2 n \log\log n)$ [15] and $O(m^2 l^2 n \log^3 \ n \log\log n)$ [16], respectively. Despite the latter two approaches employ fast multiplication, they become effective only if the polynomial degrees are sufficiently large. Since this work considers moderate also practical size RS codes, the polynomial degrees have not reached a level that can be facilitated. In fact, the re-encoding transform further reduces the polynomial degrees into only tens. Therefore, in this work, we apply the Mulders-Storjohann (MS) algorithm [14].

III. The ACD-MM Algorithm

This section reviews the ACD-MM algorithm [34], which can be regarded as a predecessor of our proposals. They adopt the same test-vectors formulation and re-encoding transform.

A. Test-Vectors Formulation

Assume codeword $\underline{c}$ 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 symbol vector. A reliability matrix $\mathbf{\Pi} \in \mathbb{R}^{q\times n}$ can be obtained, whose entries are $\pi_{ij} = \Pr[c_j = \sigma_i \mid \mathsf{r}_j]$,[2] where $0 \le i \le q-1, 0 \le j \le n-1$. 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

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

where $\gamma_j \in (1, \infty)$. The decision is less reliable if γ_j is smaller, and vice versa. Sorting γ_j in a nonincreasing order yields a new symbol index sequence $j_0, j_1, \ldots, j_{n-1}$ that indicates $\gamma_{j_0} \ge \gamma_{j_1} \ge \cdots \ge \gamma_{j_{n-1}}$. Let $\Theta = \{j_0, j_1, \ldots, j_{n-\eta-1}\}$ denote the index set of the $n-\eta$ most reliable symbols. Its complementary set is $\Theta^{\mathrm{c}} = \{j_{n-\eta}, j_{n-\eta+1}, \ldots, j_{n-1}\}$. Since there are two decisions for each of the η unreliable symbols, 2^η test-vectors will be formulated. Consequently, all test-vectors can be written as

$$\underline{r}_u = (r_{j_0}^{(u)}, r_{j_1}^{(u)}, \ldots, r_{j_{n-\eta-1}}^{(u)}, r_{j_{n-\eta}}^{(u)}, \ldots, r_{j_{n-1}}^{(u)}), \tag{5}$$

where $u = 1, 2, \ldots, 2^\eta$ is the test-vector index and

$$r_j^{(u)} = \begin{cases} r_j^{\mathrm{I}}, & \text{if } j \in \Theta, \\ r_j^{\mathrm{I}} \text{ or } r_j^{\mathrm{II}}, & \text{if } j \in \Theta^{\mathrm{c}}. \end{cases} \tag{6}$$

Note that the reliability metric can also be defined as either the smallest bit-wise channel observation or the entropy of each received symbol. Our research shows that compared with the above symbol-wise metric, identifying the unreliable symbols using the bit-wise reliability results in the same decoding performance, while using the entropy metric shows a slight performance degradation. One may also consider more decisions for c_j. This will gain performance but with a much higher decoding complexity. Our research also shows that initiating more unreliable symbols outweighs making more decisions for fewer unreliable positions. For the proposed Chase decoding style, it is important to identify the erroneous positions and classify them unreliable so that the errors can be removed by alternating the decisions. Finding a better way to define the unreliable symbols remains to be our pursuit.

B. Re-Encoding Transform

Re-encoding further transforms the test-vectors [23]. Let $\eta \le n-k$ to ensure all test-vectors would share at least k common symbols $r_{j_0}^{\mathrm{I}}, r_{j_1}^{\mathrm{I}}, \ldots, r_{j_{k-1}}^{\mathrm{I}}$. Let $\Psi = \{j_0, j_1, \ldots, j_{k-1}\}$ denote the index set of these k most reliable symbols. Hence,

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

$\Psi^{\mathrm{c}} = \{j_k, j_{k+1}, \ldots, j_{n-1}\}$. Let $h_j = r_j^{\mathrm{I}}, \forall j \in \Psi$, the codeword $\underline{h} = (h_0, h_1, \ldots, h_{n-1})$ can be generated by Forney's erasure-only decoding algorithm [37]. All test-vectors $\underline{r}_u$ are then transformed by

$$\underline{r}_u \mapsto \underline{z}_u : z_j^{(u)} = r_j^{(u)} - h_j, \quad \forall j. \tag{7}$$

Consequently, the transformed test-vectors become

$$\underline{z}_u = (0, 0, \ldots, 0, z_{j_k}^{(u)}, \ldots, z_{j_{n-1}}^{(u)}). \tag{8}$$

C. Basis Construction and Reduction

For all test-vectors $\underline{z}_u$, the polynomial (2) is redefined as

$$R_u(x) = \sum_{j=0}^{n-1} z_j^{(u)} T_j(x). \tag{9}$$

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

$$V(x) = \prod_{j \in \Psi} (x - \alpha_j) \tag{10}$$

becomes the GCD for both $G(x)$ of (1) and the above $R_u(x)$. Therefore, two new module seeds are defined as [34]

$$\tilde{G}(x) = \frac{G(x)}{V(x)} = \prod_{j \in \Psi^{\mathrm{c}}} (x - \alpha_j) \tag{11}$$

and

$$\tilde{R}_u(x) = \frac{R_u(x)}{V(x)} = \sum_{j \in \Psi^{\mathrm{c}}} z_j^{(u)} \tilde{T}_j(x), \tag{12}$$

where

$$\tilde{T}_j(x) = \frac{\prod_{j' \in \Psi^{\mathrm{c}}, j' \neq j} (x - \alpha_{j'})}{\prod_{j'=0, j' \neq j}^{n-1} (\alpha_j - \alpha_{j'})}.$$

Now the isomorphic module $\tilde{\mathcal{M}}_u$ for each test-vector $\underline{z}_u$ can be generated by

$$\tilde{P}_t(x, y) = \tilde{G}(x)^{m-t} (y - \tilde{R}_u(x))^t, \quad \text{if } 0 \leq t \leq m, \tag{13}$$

$$\tilde{P}_t(x, y) = (yV(x))^{t-m} (y - \tilde{R}_u(x))^m, \quad \text{if } m < t \leq l. \tag{14}$$

They form a basis $\tilde{\mathcal{B}}_u$ of $\tilde{\mathcal{M}}_u$, which will be reduced into a Gröbner basis for finding the interpolation polynomial $Q_u(x, y)$ and recovering the message [34].

IV. The LCC-BR Algorithm

This section introduces the LCC-BR algorithm, in which we set $m = l = 1$. Under this configuration, the MS algorithm is equivalent to the conventional extended Euclidean algorithm [3], which will be shown in Remark 1. For high rate RS codes, increasing m (or l) cannot improve the error-correction capability of each Chase decoding event. Since high rate codes are more preferable in practice, setting $m = l = 1$ does not limit the proposed algorithms' error-correction capability in action. The LCC-BR algorithm consists of test-vectors formulation, re-encoding transform, common basis construction and individual basis construction. In particular, the first two steps are the same as the ACD-MM algorithm [34]. Hence, we begin with introducing the common basis construction.

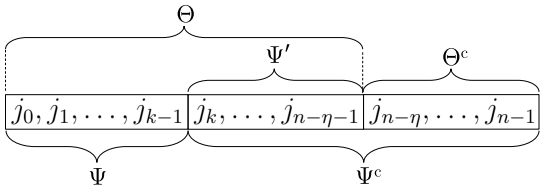


Fig. 1. Index sets of the LCC-BR algorithm.

A. Common Basis Construction

From (6), it can be observed that the common interpolation points of the 2^η test-vectors are $(\alpha_j, r_j^{\mathrm{I}}), \forall j \in \Theta$. After the re-encoding transform, they become $(\alpha_j, z_j^{\mathrm{I}})$, where $z_j^{\mathrm{I}} = r_j^{\mathrm{I}} - h_j$. Note that $z_j^{\mathrm{I}} = 0, \forall j \in \Psi$. Let $\Psi' = \Psi^{\mathrm{c}} \setminus \Theta^{\mathrm{c}} = \{j_k, j_{k+1}, \ldots, j_{n-\eta-1}\}$. For convenience, the above mentioned index sets are illustrated in Fig. 1. We define a common test-vector

$$\underline{z}_0 = (z_0^{(0)}, z_1^{(0)}, \ldots, z_{n-1}^{(0)}), \tag{15}$$

where $z_j^{(0)} = z_j^{\mathrm{I}}, \forall j \in \Psi'$ and $z_j^{(0)} = 0, \forall j \in \Psi \cup \Theta^{\mathrm{c}}$. Since $m = l = 1$, the module generators (13) and (14) defined by $\underline{z}_0$ can be simplified into

$$\tilde{P}_{0,0}(x, y) = \tilde{G}(x), \tag{16}$$

$$\tilde{P}_{0,1}(x, y) = y - \tilde{R}_0(x), \tag{17}$$

where

$$\tilde{R}_0(x) = \sum_{j \in \Psi^{\mathrm{c}}} z_j^{(0)} \tilde{T}_j(x) = \sum_{j \in \Psi'} z_j^{(0)} \tilde{T}_j(x).$$

The above two polynomials generate an isomorphic module $\tilde{\mathcal{M}}_0$ for the points $(\alpha_j, z_j^{(0)}), \forall j \in \Psi'$, forming the common basis $\tilde{\mathcal{B}}_0$. The MS algorithm [14] that performs $\mathbb{F}_q[x]$-linear combination will reduce $\tilde{\mathcal{B}}_0$ into a Gröbner basis $\tilde{\mathcal{B}}_0'$, where its entries are denoted as $\tilde{P}'_{0,0}(x, y)$ and $\tilde{P}'_{0,1}(x, y)$. Based on Lemma 2, they can be written as

$$\tilde{P}'_{0,0}(x, y) = p_{00}(x)\tilde{G}(x) + p_{01}(x)(y - \tilde{R}_0(x)), \tag{18}$$

$$\tilde{P}'_{0,1}(x, y) = p_{10}(x)\tilde{G}(x) + p_{11}(x)(y - \tilde{R}_0(x)), \tag{19}$$

where $p_{00}(x), p_{01}(x), p_{10}(x), p_{11}(x) \in \mathbb{F}_q[x]$. With the re-encoding transform, polynomials are organized by the $(1, -1)$-revlex order [24]. Based on Lemma 3, we have $\deg_y \mathrm{LM}_{1,-1}(\tilde{P}'_{0,0}(x, y)) \neq \deg_y \mathrm{LM}_{1,-1}(\tilde{P}'_{0,1}(x, y))$. These two polynomials will be utilized by the following 2^η individual basis constructions.

Remark 1: When $m = l = 1$, the MS algorithm is equivalent to the extended Euclidean algorithm [3]. Assume that

$$g_0(x, y) = g_0^{(0)}(x) + g_0^{(1)}(x)y$$

$$g_1(x, y) = g_1^{(0)}(x) + g_1^{(1)}(x)y$$

generate the module $\mathcal{M}$. Further assume that $g_0(x, y)$ has the leading term (LT) in $g_0^{(0)}(x)$ and $\deg g_0^{(0)}(x) > \deg g_1^{(0)}(x)$. If $g_1(x, y)$ also has the leading term in $g_1^{(0)}(x)$, then

$\mathcal{B} = \{g_0(x,y), g_1(x,y)\}$ is not a Gröbner basis. For the MS algorithm [14], it will perform

$$g_0(x,y) \leftarrow g_0(x,y) - \frac{\mathrm{LT}(g_0^{(0)}(x))}{\mathrm{LT}(g_1^{(0)}(x))} g_1(x,y)$$

to update $g_0(x,y)$. This update is equivalent to calculating

$$g_0^{(0)}(x) = \tilde{q}(x) g_1^{(0)}(x) + \tilde{r}(x),$$

where $\deg \tilde{r}(x) < \deg g_1^{(0)}(x)$, and updating $g_0(x,y)$ by

$$\begin{aligned} g_0(x,y) &\leftarrow g_0(x,y) - \tilde{q}(x) g_1(x,y) \\ &= \tilde{r}(x) + (g_0^{(1)}(x) - \tilde{q}(x) g_1^{(1)}(x)) y, \end{aligned}$$

i.e., $g_0^{(0)}(x) \leftarrow \tilde{r}(x)$ and $g_0^{(1)}(x) \leftarrow (g_0^{(1)}(x) - \tilde{q}(x) g_1^{(1)}(x))$. Note that $\tilde{r}(x)$ and $\tilde{q}(x)$ can be calculated by the extended Euclidean algorithm [3]. Therefore, the MS algorithm is equivalent to the extended Euclidean algorithm in solving the basis reduction problem of our work. Their complexity would also be the same if the calculation of $\tilde{r}(x)$ and $\tilde{q}(x)$ uses the naive polynomial division. When their degrees are sufficiently large, a fast extended Euclidean algorithm based on the divide-and-conquer concept can be used to facilitate the process [38].

B. Individual Basis Construction

Based on $\tilde{\mathcal{B}}_0'$, the BR interpolation will be completed by performing the individual basis construction and reduction for each transformed test-vector $\underline{z}_u$. Since $z_j^{(u)} = z_j^{(0)}, \forall j \in \Psi'$, and $\Psi^{\mathrm{c}} = \Psi' \cup \Theta^{\mathrm{c}}$, the polynomial (12) can be rewritten as

$$\tilde{R}_u(x) = \tilde{R}_0(x) + \tilde{\Upsilon}_u(x),$$

where

$$\tilde{\Upsilon}_u(x) = \sum_{j \in \Theta^{\mathrm{c}}} z_j^{(u)} \tilde{T}_j(x).$$

Therefore, based on (18) and (19), for each test-vector $\underline{z}_u$, we have

$$\begin{aligned} \tilde{P}_{u,t}(x,y) &= p_{t0}(x)\tilde{G}(x) + p_{t1}(x)(y - \tilde{R}_u(x)) \\ &= p_{t0}(x)\tilde{G}(x) + p_{t1}(x)(y - \tilde{R}_0(x) - \tilde{\Upsilon}_u(x)) \\ &= \tilde{P}'_{0,t}(x,y) - p_{t1}(x)\tilde{\Upsilon}_u(x), \end{aligned} \tag{20}$$

where $t = 0, 1$. Polynomials $\tilde{P}_{u,0}(x,y)$ and $\tilde{P}_{u,1}(x,y)$ form the basis $\tilde{\mathcal{B}}_u$. It can be seen that all $\tilde{\mathcal{B}}_u$ are constructed using the previously reduced common basis $\tilde{\mathcal{B}}_0'$, eliminating the redundant computation in decoding all test-vectors. Afterwards, the MS algorithm will reduce each basis $\tilde{\mathcal{B}}_u$ into its Gröbner basis $\tilde{\mathcal{B}}_u'$ which contains $\tilde{P}'_{u,0}(x,y)$ and $\tilde{P}'_{u,1}(x,y)$. Determine $\tilde{Q}_u(x,y)$ by

$$\tilde{Q}_u(x,y) = \min\{\tilde{P}'_{u,0}(x,y), \tilde{P}'_{u,1}(x,y)\}. \tag{21}$$

Since $\tilde{Q}_u(x,y) = \tilde{Q}_u^{(0)}(x) + \tilde{Q}_u^{(1)}(x) y$, it can be restored into the interpolation polynomial $Q_u(x,y)$ by

$$Q_u(x,y) = V(x)\tilde{Q}_u^{(0)}(x) + \tilde{Q}_u^{(1)}(x) y. \tag{22}$$

Note that $Q_u(x,y)$ interpolates the points $(\alpha_0, z_0^{(u)})$, $(\alpha_1, z_1^{(u)}), \ldots, (\alpha_{n-1}, z_{n-1}^{(u)})$ with a multiplicity of one. If the decoding estimation $\tilde{f}_u(x)$ satisfies $Q_u(x, \tilde{f}_u(x)) = 0$, i.e.,

$$Q_u(x, \tilde{f}_u(x)) = V(x)\tilde{Q}_u^{(0)}(x) + \tilde{Q}_u^{(1)}(x)\tilde{f}_u(x) = 0, \tag{23}$$

Algorithm 1 The LCC-BR Algorithm

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

```
Formulate 2^η test-vectors r_u as in (5);
Perform the re-encoding transform to yield z_u as in (8);
Construct B̃_0 as in (16) (17) and reduce it into B̃'_0;
For each transformed test-vector z_u do
    Construct B̃_u as in (20);
    Perform the MS algorithm to reduce B̃_u into B̃'_u;
    Determine Q_u(x,y) as in (21) (22);
    Decode f̃_u(x) as in (24) and estimate f̂_u(x);
End for
```

$\tilde{f}_u(x)$ can be determined by

$$\tilde{f}_u(x) = -\frac{V(x)\tilde{Q}_u^{(0)}(x)}{\tilde{Q}_u^{(1)}(x)}, \tag{24}$$

and the estimated codeword is $\hat{\underline{c}}_u = (\hat{c}_0^{(u)}, \hat{c}_1^{(u)}, \ldots, \hat{c}_{n-1}^{(u)})$, where $\hat{c}_j^{(u)} = \tilde{f}_u(\alpha_j) + h_j, \forall j$. Its corresponding message polynomial $\hat{f}_u(x)$ can be determined by the following discrete Fourier transform (DFT) [39]. By presenting $\hat{\underline{c}}_u$ as $\hat{c}_u(x) = \hat{c}_0^{(u)} + \hat{c}_1^{(u)} x + \cdots + \hat{c}_{n-1}^{(u)} x^{n-1}$, the coefficients of the estimated message polynomial $\hat{f}_u(x) = \hat{f}_0^{(u)} + \hat{f}_1^{(u)} x + \cdots + \hat{f}_{n-1}^{(u)} x^{n-1}$ can be determined by $\hat{f}_j^{(u)} = n^{-1} \cdot \hat{c}_u(\alpha_j^{-1}), \forall j$. Note that $n^{-1} = 1$ in the binary extension field and $\hat{f}_j^{(u)} = 0$ for $k \le j \le n-1$. If $V(x)\tilde{Q}_u^{(0)}(x)$ cannot be divided by $\tilde{Q}_u^{(1)}(x)$, the decoding of the test-vector $\underline{z}_u$ fails. After decoding all 2^η test-vectors, the message $\hat{f}_u(x)$ whose corresponding codeword $\hat{\underline{c}}_u$ yields the minimum Euclidean distance (after modulation) to the received symbol vector $\underline{\mathrm{r}}$ will be chosen as the output. The LCC-BR algorithm is summarized in Algorithm 1.

Unlike the LCC algorithm [30], the LCC-BR algorithm can perform the decoding for each test-vector in parallel, offering a low decoding latency. Furthermore, Fig. 2 shows the difference between the ACD-MM [34] and the LCC-BR algorithms. It can be seen that the proposed LCC-BR algorithm improves upon the existing ACD-MM algorithm by utilizing the similarity among all 2^η test-vectors to eliminate the redundant computation in the basis construction and reduction.

V. The Progressive LCC-BR Algorithm

The above description shows the LCC-BR algorithm needs to decode 2^η test-vectors. However, if the channel condition improves, the received information becomes less corruptive. It will not be necessary to decode all test-vectors since the intended message can be decoded by processing fewer test-vectors. Therefore, we design the progressive LCC-BR algorithm, in which the test-vectors are decoded in a sequential manner. Once the intended message is decoded, the decoding terminates. During the progressive decoding, the BR interpolation results of the current test-vector is utilized by the following test-vector, saving the decoding computation. Unlike the progressive LCC algorithm [32], the proposed algorithm

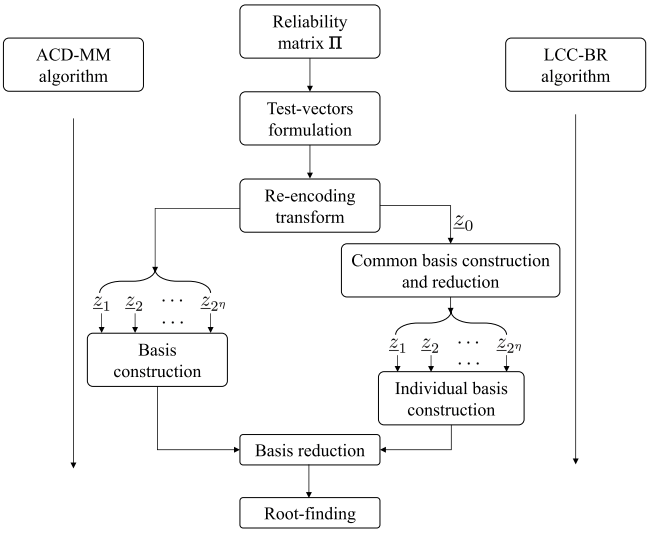


Fig. 2. Diagram of the ACD-MM and the LCC-BR algorithms.

does not have additional memory requirement. The progressive decoding will first sort the test-vectors such that the one with a higher potential of yielding the intended message will be decoded earlier [32].

A. Ordering of Test-Vectors

Given a test-vector $\underline{r}_u = (r_0^{(u)}, r_1^{(u)}, \ldots, r_{n-1}^{(u)})$, its reliability can be defined as [32]

$$\Omega_u = \prod_{j=0}^{n-1} \pi_{i_j^{(u)} j},$$

where $i_j^{(u)} = \text{index}\{\sigma_i \mid \sigma_i = r_j^{(u)}\}$. A test-vector with a larger Ω_u value is regarded to be more reliable and it should be decoded earlier. Since all test-vectors share the common symbols r_j^{I}, where $j \in \Theta$, the reliability function can be further simplified into

$$\tilde{\Omega}_u = \prod_{j \in \Theta^{\text{c}}} \pi_{i_j^{(u)} j}. \tag{25}$$

By sorting the reliability of 2^η test-vectors in a descending order, a refreshed index sequence $u_1, u_2, \ldots, u_{2^\eta}$ can be yielded, which indicates $\tilde{\Omega}_{u_1} > \tilde{\Omega}_{u_2} > \cdots > \tilde{\Omega}_{u_{2^\eta}}$. Note that the first decoded test-vector is the hard-decision received word, i.e., $\underline{r}_{u_1} = \underline{\omega}$. The progressive algorithm will decode the test-vectors sequentially until a codeword that satisfies the ML criterion [33] is found. It has been shown in [32] that such an ordering enables the decoding to be terminated earlier, minimizing the message recovery complexity.

B. The Algorithm

Let $\Lambda_{u_\tau} = \{j \mid z_j^{(u_\tau)} \neq z_j^{(u_{\tau+1})}, j \in \Theta^{\text{c}}\}$ denote the index set of the different symbols between $\underline{z}_{u_\tau}$ and $\underline{z}_{u_{\tau+1}}$, where $\tau = 1, 2, \ldots, 2^\eta$. Note that $\Lambda_{u_{2^\eta}} = \emptyset$. Since $z_j^{(u_\tau)} - z_j^{(u_{\tau+1})} = (r_j^{(u_\tau)} - h_j) - (r_j^{(u_{\tau+1})} - h_j) = r_j^{(u_\tau)} - r_j^{(u_{\tau+1})}$, $\forall j$, $z_j^{(u_\tau)} \neq z_j^{(u_{\tau+1})}$ implies $r_j^{(u_\tau)} \neq r_j^{(u_{\tau+1})}$. Therefore, Λ_{u_τ} can also be denoted as

$$\Lambda_{u_\tau} = \{j \mid r_j^{(u_\tau)} \neq r_j^{(u_{\tau+1})}, j \in \Theta^{\text{c}}\}. \tag{26}$$

The progressive LCC-BR algorithm is described as follows. Unlike the LCC-BR algorithm, the common basis does not need to be constructed. At the beginning, a basis $\tilde{\mathcal{B}}_{u_1}$ for the test-vector $\underline{z}_{u_1}$ will be constructed by

$$\tilde{P}_{u_1,0}(x, y) = \tilde{G}(x), \tag{27}$$
$$\tilde{P}_{u_1,1}(x, y) = y - \tilde{R}_{u_1}(x). \tag{28}$$

The MS algorithm will reduce $\tilde{\mathcal{B}}_{u_1}$ into a Gröbner basis $\tilde{\mathcal{B}}'_{u_1}$ which contains the polynomials $\tilde{P}'_{u_1,0}(x, y)$ and $\tilde{P}'_{u_1,1}(x, y)$. The minimum one will be chosen as $\tilde{Q}_{u_1}(x, y)$. The estimated message $\hat{f}_{u_1}(x)$ can be determined as in (24) and the DFT. If its corresponding codeword $\underline{\hat{c}}_{u_1}$ satisfies the ML criterion [33], the decoding terminates and outputs $\hat{f}_{u_1}(x)$. Otherwise, the decoding continues to decode the test-vector $\underline{z}_{u_2}$. Based on Lemma 2, the polynomials $\tilde{P}'_{u_1,0}(x, y)$ and $\tilde{P}'_{u_1,1}(x, y)$ can be written as

$$\tilde{P}'_{u_1,0}(x, y) = p_{u_1,00}(x)\tilde{G}(x) + p_{u_1,01}(x)(y - \tilde{R}_{u_1}(x)),$$
$$\tilde{P}'_{u_1,1}(x, y) = p_{u_1,10}(x)\tilde{G}(x) + p_{u_1,11}(x)(y - \tilde{R}_{u_1}(x)),$$

where $p_{u_1,00}(x), p_{u_1,01}(x), p_{u_1,10}(x), p_{u_1,11}(x) \in \mathbb{F}_q[x]$. The difference between $\underline{z}_{u_1}$ and $\underline{z}_{u_2}$ is used to formulate the polynomial $\tilde{R}_{u_2}(x)$ as

$$\tilde{R}_{u_2}(x) = \tilde{R}_{u_1}(x) + W_{u_1}(x),$$

where $W_{u_1}(x) = \sum_{j \in \Lambda_{u_1}} (z_j^{(u_2)} - z_j^{(u_1)})\tilde{T}_j(x) = \sum_{j \in \Lambda_{u_1}} (r_j^{(u_2)} - r_j^{(u_1)})\tilde{T}_j(x)$. A basis $\tilde{\mathcal{B}}_{u_2}$ for the test-vector $\underline{z}_{u_2}$ can be constructed by

$$\begin{aligned}
&\tilde{P}_{u_2,t}(x, y) \\
&\quad = p_{u_1,t0}(x)\tilde{G}(x) + p_{u_1,t1}(x)(y - \tilde{R}_{u_2}(x)) \\
&\quad = p_{u_1,t0}(x)\tilde{G}(x) + p_{u_1,t1}(x)(y - \tilde{R}_{u_1}(x) - W_{u_1}(x)) \\
&\quad = \tilde{P}'_{u_1,t}(x, y) - p_{u_1,t1}(x)W_{u_1}(x),
\end{aligned}$$

where $t = 0, 1$. It can be seen that the polynomials $\tilde{P}_{u_2,0}(x, y)$ and $\tilde{P}_{u_2,1}(x, y)$ are constructed based on $\tilde{P}'_{u_1,0}(x, y)$ and $\tilde{P}'_{u_1,1}(x, y)$. Moreover, the re-encoding transform should only be performed for the test-vector $\underline{r}_{u_1}$. The basis of other test-vectors does not rely on their transform as $W_{u_1}(x)$ indicates. The MS algorithm reduces $\tilde{\mathcal{B}}_{u_2}$ into a Gröbner basis $\tilde{\mathcal{B}}'_{u_2}$ in which the minimum candidate will be restored into the interpolation polynomial $Q_{u_2}(x, y)$.

In general, if the ML codeword cannot be retrieved from decoding the test-vector $\underline{z}_{u_{\tau-1}}$ ($\tau \geq 2$), the next test-vector $\underline{z}_{u_\tau}$ needs to be decoded. Based on the above derivation, a basis $\tilde{\mathcal{B}}_{u_\tau}$ for $\underline{z}_{u_\tau}$ can be constructed by

$$\tilde{P}_{u_\tau,0}(x, y) = \tilde{P}'_{u_{\tau-1},0}(x, y) - p_{u_{\tau-1},01}(x)W_{u_{\tau-1}}(x), \tag{29}$$
$$\tilde{P}_{u_\tau,1}(x, y) = \tilde{P}'_{u_{\tau-1},1}(x, y) - p_{u_{\tau-1},11}(x)W_{u_{\tau-1}}(x), \tag{30}$$

where $\tilde{P}'_{u_{\tau-1},0}(x, y)$, $\tilde{P}'_{u_{\tau-1},1}(x, y)$, $p_{u_{\tau-1},01}(x)$ and $p_{u_{\tau-1},11}(x)$ are obtained from the decoding of $\underline{z}_{u_{\tau-1}}$,

and

$$W_{u_{\tau-1}}(x) = \sum_{j\in\Lambda_{u_{\tau-1}}} (z_j^{(u_\tau)} - z_j^{(u_{\tau-1})})\tilde{T}_j(x) = \sum_{j\in\Lambda_{u_{\tau-1}}} (r_j^{(u_\tau)} - r_j^{(u_{\tau-1})})\tilde{T}_j(x). \quad (31)$$

The MS algorithm will reduce $\tilde{\mathcal{B}}_{u_\tau}$ into its Gröbner basis $\tilde{\mathcal{B}}'_{u_\tau}$, in which the minimum candidate will be restored as in (22) and its y-root is determined as in (24). If it produces a codeword $\underline{\hat{c}}_{u_\tau}$ that satisfies the ML criterion, the decoding will terminate and output the estimated message $\hat{f}_{u_\tau}(x)$. Otherwise, the decoding continues to decode the test-vector $\underline{z}_{u_{\tau+1}}$. If the decoding of all 2^η test-vectors cannot produce an ML codeword, the decoding terminates with a failure.

Remark 2: Equations (28) and (31) reveal that the re-encoding transform only needs to be performed once on the test-vector $\underline{r}_{u_1}$.

Remark 3: It can be observed from (29) and (30) that the progressive LCC-BR algorithm does not need to memorize the intermediate decoding information, eliminating the memory cost of the original progressive algorithm [32].

VI. Decoding Complexity and Latency

This section analyzes the decoding complexity and latency of the proposed algorithms. They are measured as the number of finite field multiplications[3] and the running time required to decode a codeword, respectively.

A. The LCC-BR Algorithm

Based on the above descriptions, the LCC-BR algorithm includes re-encoding transform, common basis construction, individual basis construction and root-finding. Their complexity are denoted as $\mathcal{C}_{\text{re}}$, $\mathcal{C}_{\text{com}}$, $\mathcal{C}_{\text{ind}}$ and $\mathcal{C}_{\text{rf}}$, respectively. They will be characterized so that an overview of the algorithms complexity can be obtained.

Lemma 4 [30]: Complexity of the re-encoding transform is $\mathcal{C}_{\text{re}} = 4(n-k)^2$.

Lemma 5: Complexity of the common basis construction is $\mathcal{C}_{\text{com}} = \frac{1}{2}(n-k)^2(n-k+7)$.

Proof: The common basis construction complexity is measured by the number of multiplications in computing the generators (16) and (17), and the Gröbner basis $\mathcal{B}'_0$. As $|\Psi^{\text{c}}| = n-k$, computing $\tilde{G}(x)$ requires $\frac{1}{2}(n-k)(n-k+1)$ multiplications. Note that there are $n-k$ polynomials $\tilde{T}_j(x)$ and computing each one requires $\frac{1}{2}(n-k)(n-k-1)$ multiplications. Hence, the multiplications in computing all $\tilde{T}_j(x)$ are $\frac{1}{2}(n-k)^2(n-k-1)$. Since $|\Psi'| = n-k-\eta$ and $\deg \tilde{T}_j(x) = n-k-1$, further computing $\tilde{R}_0(x)$ requires $(n-k-1)(n-k-\eta)$ multiplications. The MS algorithm needs at most $(n-k+1)$ $\mathbb{F}_g[x]$-linear combinations to reduce $\mathcal{B}_0$ into $\mathcal{B}'_0$ [28]. Since $\deg \tilde{G}(x) = n-k$ and $\deg \tilde{R}_0(x) = n-k-1$, the common basis reduction requires at most $2(n-k)(n-k+1)$ multiplications. With $\eta \le n-k$, the above analysis shows that complexity of the common basis construction can be approximated to $\mathcal{C}_{\text{com}} = \frac{1}{2}(n-k)^2(n-k+7)$. ■

Note that with the re-encoding transform, the index set Ψ^{c} varies in each decoding event. Therefore, $\tilde{G}(x)$ and all $\tilde{T}_j(x)$ cannot be computed offline. The above two steps of re-encoding transform and common basis construction are performed once, producing an outcome shared by all 2^η test-vectors.

Lemma 6: Complexity of the individual basis construction is $\mathcal{C}_{\text{ind}} = 2^\eta \cdot 5(n-k)^2$.

Proof: The individual basis construction complexity is determined by the complexity in computing the generators (20) and the Gröbner basis $\mathcal{B}'_u$. Since $|\Theta^{\text{c}}| = \eta$, computing $\tilde{\Upsilon}_u(x)$ requires $\eta(n-k-1)$ multiplications. Since $\deg p_{01}(x) \le \frac{1}{2}(n-k)$, $\deg p_{11}(x) \le \frac{1}{2}(n-k)$ and $\deg \tilde{\Upsilon}_u(x) = n-k-1$, computing the polynomials $\tilde{P}_{u,0}(x,y)$ and $\tilde{P}_{u,1}(x,y)$ requires at most $\frac{1}{2}(n-k)(n-k-1)\cdot 2 = (n-k)(n-k-1)$ multiplications. Furthermore, since $\deg_x \tilde{P}_{u,0}(x,y) \le \frac{3}{2}(n-k)$ and $\deg_x \tilde{P}_{u,1}(x,y) \le \frac{3}{2}(n-k)$, the individual basis reduction requires at most $3(n-k+1)(n-k)$ multiplications. With 2^η test-vectors, complexity of the individual basis construction can be approximated to $\mathcal{C}_{\text{ind}} = 2^\eta \cdot 5(n-k)^2$. ■

Lemma 7: Complexity of the root-finding is $\mathcal{C}_{\text{rf}} = 2^\eta \cdot k(n-k)$.

Proof: The root-finding complexity is determined by the complexity in computing (24). Since $\deg V(x) = k$ and $\deg \tilde{Q}_u^{(0)}(x) \le \frac{1}{2}(n-k)$, computing $V(x)\tilde{Q}_u^{(0)}(x)$ requires $\frac{1}{2}k(n-k)$ multiplications. Since $\deg V(x)\tilde{Q}_u^{(0)}(x) \le \frac{1}{2}(n+k)$, the division between $V(x)\tilde{Q}_u^{(0)}(x)$ and $\tilde{Q}_u^{(1)}(x)$ requires at most $k(\frac{1}{2}(n-k)+1)$ multiplications. Therefore, complexity of the root-finding can be approximated to $\mathcal{C}_{\text{rf}} = 2^\eta \cdot k(n-k)$. ■

The above complexity characterizations show that the overall complexity of the LCC-BR algorithm is $\mathcal{C}_{\text{LCC-BR}} = \mathcal{C}_{\text{re}} + \mathcal{C}_{\text{com}} + \mathcal{C}_{\text{ind}} + \mathcal{C}_{\text{rf}}$. When η is sufficiently large, the LCC-BR complexity is dominated by the individual basis construction and the root-finding. Asymptotically, it is $O(2^\eta(n-k)^2)$. Otherwise, it will be dominated by the common basis construction with an asymptotic characterization of $O((n-k)^3)$. These asymptotic characterizations show that the LCC-BR algorithm will be more effective for high rate codes. Table I shows our numerical results in decoding various RS codes. They were obtained by simulating the LCC-BR algorithm, during which the average number of finite field multiplications in decoding a codeword is measured. We verify our analysis by seeing whether the simulation results yield the same magnitude as the analytical expressions. For example, when $\eta = 6$, the analytical complexity is $\mathcal{C}_{\text{LCC-BR}} = 4.15 \times 10^5$ for the (63, 31) RS code, while the simulation shows 2.44×10^5.

TABLE I
The LCC-BR Complexity in Decoding Various RS Codes

η	(63, 15)	(63, 31)	(63, 55)
2	8.24×10^4	3.75×10^4	1.31×10^4
4	1.22×10^5	7.81×10^4	4.47×10^4
6	2.88×10^5	2.44×10^5	1.50×10^5

[3]Finite field multiplications dominate the computation in the decoding.

TABLE II
COMPLEXITY COMPARISON IN DECODING THE (63, 47) RS CODE

η	LCC [30]	BF-LCC [31]	ACD-MM [34]	LCC-BR
2	1.65×10^4	2.20×10^4	2.51×10^4	1.84×10^4
4	5.61×10^4	6.80×10^4	7.44×10^4	5.82×10^4
6	2.12×10^5	2.54×10^5	2.73×10^5	2.16×10^5

Their discrepancy comes from the fact that we only consider the dominant terms in the analytical expressions. Table I also shows that the LCC-BR algorithm yields a lower complexity for high rate codes, which will be more welcome in practice.

Table II compares the complexity of the LCC-BR algorithm with several existing Chase decoding algorithms. The (63, 47) RS code was used. Note that all Chase decoding algorithms employ the same test-vectors formulation. Both the LCC [30] and the BF-LCC [31] algorithms employ Kötter's interpolation. It can be seen that the complexity of all algorithms have the same magnitude, where the LCC-BR algorithm has a lower complexity than the BF-LCC algorithm. By eliminating the redundant BR computation of the ACD-MM algorithm, the proposed LCC-BR algorithm yields a lower complexity. It can also be seen that the LCC-BR algorithm has a slightly higher complexity than the LCC algorithm. Despite the BR interpolation is less complex than Kötter's interpolation, the LCC algorithm performs the interpolation of all test-vectors in a binary-tree growing fashion, granting it a low-complexity feature. However, the LCC-BR algorithm has a significant advantage in decoding latency, which is shown below.

Since the LCC-BR algorithm can perform its individual basis construction and root-finding in parallel, the re-encoding transform and the common basis construction dominate the running time. Therefore, its decoding latency does not vary remarkably with different η values and it will be defined by that of decoding a single test-vector. Lemmas 4 and 5 show that high rate codes have a lower LCC-BR decoding complexity. Table III shows the complexity and latency (in ms) in decoding two RS codes defined over $\mathbb{F}_{256}$. These results were obtained by simulating the algorithms in C based on the Intel core i5-4260U CPU and the macOS mojave operating platform. For the rate half (255, 127) RS code, the LCC and the BF-LCC algorithms yield a lower complexity than the LCC-BR algorithm. The situation reverses in decoding the (255, 239) RS code. In contrast to the LCC-BR algorithm, running time of the LCC and the BF-LCC algorithms increase as the η value enlarges. The LCC-BR algorithm maintains a rather stable latency thanks to its parallel decoding feature. Table III also shows that the LCC-BR decoding latency of the (255, 239) RS code is smaller than that of the (255, 127) RS code, validating its effectiveness for high rate codes. In comparison with the LCC and the BF-LCC algorithms, the LCC-BR algorithm shows its advantage on the decoding latency. Its complexity advantage also emerges for high rate codes. This again demonstrates the proposal's practical merit.

B. The Progressive LCC-BR Algorithm

In order to show more insights of the channel dependent feature of the progressive LCC-BR algorithm, we measure the average decoding complexity over multiple decoding events at a certain SNR. Let N_{avg} denote the average number of the decoded test-vectors in each decoding event, the average complexity of the progressive decoding will be $\mathcal{C}_{\text{Prog.}} = \mathcal{C}_{\text{re}} + \mathcal{C}_{\text{com}} + \frac{N_{\text{avg}}}{2^\eta}(\mathcal{C}_{\text{ind}} + \mathcal{C}_{\text{rf}})$. When all decoding events are terminated without producing an ML codeword, i.e., $N_{\text{avg}} = 2^\eta$, $\mathcal{C}_{\text{Prog.}} = \mathcal{C}_{\text{LCC-BR}}$. Therefore, the complexity advantage of the progressive decoding will become obvious when the channel condition improves, so that the decoding can be terminated earlier.

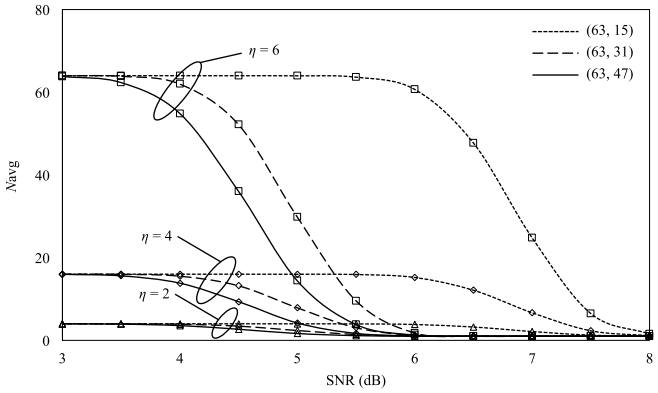


Fig. 3. Average number of decoded test-vectors in the progressive decoding.

Fig. 3 shows the statistics of N_{avg} in decoding various RS codes. The results were obtained by running 10 000 decoding events at each SNR over the additive white Gaussian noise (AWGN) channel using BPSK. As the SNR increases, fewer test-vectors are decoded, leading to a lower complexity. With a sufficiently high SNR, most of the decoding events are terminated with decoding one test-vector, i.e., $N_{\text{avg}} = 1$. The progressive LCC-BR complexity converges to the minimum level that is characterized by performing the MM based GS algorithm.

Table IV further shows the average complexity in decoding the (63, 47) RS code. For this code, complexity of the BM and the GMD algorithms are 2.42×10^3 and 2.60×10^4, respectively. It can be seen that complexity of the progressive LCC-BR algorithm decreases as the SNR increases. When the SNR is sufficiently large, e.g., SNR ≥ 6 dB, the progressive LCC-BR complexity converges to the minimum level that has the same magnitude as the BM complexity. This convergence also echoes the results of Fig. 3. Compared with Table II, the progressive LCC-BR algorithm yields a lower complexity than the LCC-BR algorithm over the whole spectrum of SNR due to its early termination feature.

VII. DECODING PERFORMANCE

This section shows the decoding performance of the proposed algorithms. They are measured as the frame error rate (FER) obtained over the AWGN channel using BPSK. Note that KV-MM and ReT-KV-MM refer to the KV algorithm and its re-encoding transformed variant whose interpolation are realized by the basis reduction approach, respectively. The Berlekamp [1], Massey [2], and the Guruswami and

TABLE III

COMPLEXITY AND LATENCY (MS) COMPARISONS BETWEEN THE LCC, THE BF-LCC AND THE LCC-BR ALGORITHMS

η	LCC [30]		BF-LCC [31]		LCC-BR	
	(255, 127) RS	(255, 239) RS	(255, 127) RS	(255, 239) RS	(255, 127) RS	(255, 239) RS
2	2.80×10^5 6.687	2.95×10^5 7.921	2.93×10^5 7.811	3.09×10^5 8.615	1.40×10^6 34.061	2.21×10^5 1.687
4	8.18×10^5 19.029	1.07×10^6 27.769	8.68×10^5 23.603	1.13×10^6 30.195	2.06×10^6 33.684	8.28×10^5 1.632
6	2.98×10^6 70.038	4.20×10^6 104.337	3.15×10^6 81.946	4.39×10^6 113.109	4.74×10^6 33.471	3.12×10^6 1.604

TABLE IV

AVERAGE COMPLEXITY OF THE PROGRESSIVE ALGORITHMS IN DECODING THE (63, 47) RS CODE

SNR (dB)	Prog. LCC-BR			Prog. KV-MM		Prog. ReT-KV-MM	
	$\eta = 2$	$\eta = 4$	$\eta = 6$	$l = 4$	$l = 8$	$l = 4$	$l = 8$
3.0	1.76×10^4	5.67×10^4	2.08×10^5	5.12×10^5	8.70×10^6	4.56×10^5	7.57×10^6
3.5	1.69×10^4	5.46×10^4	2.01×10^5	4.64×10^5	7.23×10^6	4.13×10^5	6.12×10^6
4.0	1.53×10^4	4.83×10^4	1.73×10^5	3.48×10^5	4.04×10^6	2.96×10^5	3.30×10^6
4.5	1.14×10^4	2.72×10^4	1.07×10^5	1.58×10^5	1.37×10^6	1.37×10^5	9.22×10^5
5.0	9.98×10^3	1.39×10^4	4.38×10^4	6.55×10^4	2.51×10^5	5.87×10^4	1.73×10^5
5.5	8.73×10^3	9.62×10^3	2.07×10^4	3.09×10^4	5.57×10^4	2.53×10^4	4.03×10^4
6.0	8.32×10^3	8.52×10^3	8.82×10^3	1.93×10^4	1.96×10^4	1.44×10^4	1.48×10^4
6.5	8.22×10^3	8.22×10^3	8.22×10^3	1.87×10^4	1.91×10^4	1.03×10^4	1.05×10^4
7.0	8.18×10^3	8.18×10^3	8.18×10^3	1.85×10^4	1.86×10^4	8.61×10^3	8.62×10^3

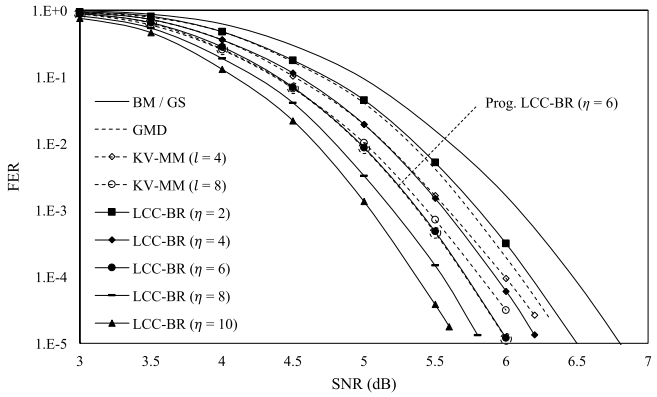


Fig. 4. Performance of the (63, 47) RS code over the AWGN channel using BPSK.

Sudan [7] and the generalized minimum-distance decoding [5] algorithms are denoted as the BM, GS and GMD algorithms, respectively.

Fig. 4 shows performance of the (63, 47) RS code. Note that with the same η, the four Chase decoding algorithms, including the LCC, the BF-LCC, the ACD-MM and the LCC-BR algorithms, yield the same performance. This is because they decode the same test-vectors. As η increases, the LCC-BR performance can be improved since more test-vectors are decoded. When $\eta = 10$, the LCC-BR algorithm outperforms the BM and the GMD algorithms with 1.1 dB and 0.7 dB coding gains at the FER of 10^{-4}, respectively. It should be pointed out that with the same η, the progressive LCC-BR algorithm maintains the decoding performance of the LCC-BR algorithm. This is evidenced by the progressive LCC-BR performance with $\eta = 6$. Revisiting the complexity results of Tables II and IV, we also know that the progressive LCC-BR complexity is lower than the LCC-BR complexity. Fig. 4 also shows performance of the KV-MM algorithm with a maximum decoding output list size of four and eight, i.e., $l = \deg_y Q(x, y) = 4$ and $l = 8$. With the same decoding parameter, the progressive KV-MM [29] and the progressive ReT-KV-MM algorithms [40] achieve the same decoding performance as the KV-MM algorithm. It can be seen that the LCC-BR algorithm with $\eta = 4$ and $\eta = 6$ outperforms the KV-MM algorithm with $l = 4$ and $l = 8$, respectively, demonstrating the proposals' competent error-correction feature. In terms of decoding complexity, the worst-case complexity of the progressive LCC-BR and the progressive KV-MM algorithms are $O(2^\eta (n-k)^2)$ and $O(l^5 n(n-k))$ [29], respectively. When the SNR is small, the progressive algorithms require a large decoding parameter with the preset η or l being approached. In that case, Table IV shows that the progressive LCC-BR algorithm will exhibit a lower decoding complexity than the progressive KV-MM algorithm. When the SNR is sufficiently large, the three progressive algorithms will decode the message with the smallest parameter. Their complexity will converge to the minimum level. Table IV shows that the progressive LCC-BR and the progressive ReT-KV-MM algorithms converge to a similar complexity level that is characterized by performing the re-encoding transform based GS algorithm with $m = 1$. They yield a lower converging level than the progressive KV-MM algorithm, thanks to the effectiveness of the re-encoding transform.

Finally, Fig. 5 shows the decoding performance of the popular (255, 239) RS code. The LCC-BR performance improves as η increases and approaches the GMD algorithm when $\eta = 2$. Compared with the BM algorithm, the LCC-BR algorithm with $\eta = 10$ yields the performance gains of 0.8 dB at the FER

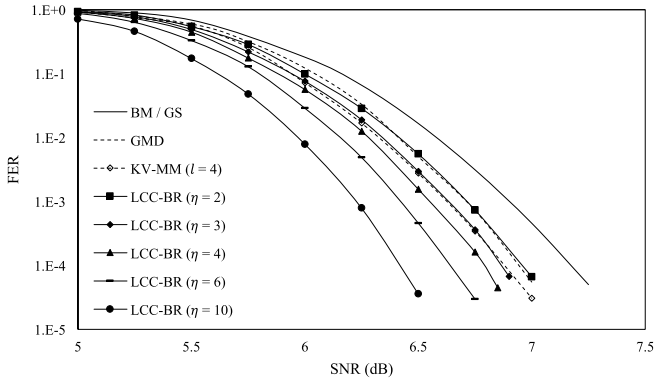


Fig. 5. Performance of the (255, 239) RS code over the AWGN channel using BPSK.

of 10^{-4}. Fig. 5 also shows that the KV-MM algorithm with $l = 4$ has a similar performance as the LCC-BR algorithm with $\eta = 3$. For this code, complexity of the ReT-KV-MM and the LCC-BR algorithms are 4.70×10^6 and 4.27×10^5, respectively. This again reveals the complexity advantage of the proposed LCC-BR algorithm.

VIII. Conclusion

This paper has proposed the BR interpolation based LCC algorithm for RS codes. Based on η unreliable symbols, 2^η test-vectors are formulated. The LCC-BR algorithm first constructs a common basis which is shared by the parallel decoding of 2^η test-vectors. It removes the redundant computation in the BR interpolation and yields a low decoding latency. The progressive LCC-BR algorithm has been further proposed to adjust the decoding computation to the quality of received information. By exploiting the difference between the adjacent test-vectors, this progressive decoding has been realized without additional memory cost. Complexity analysis has shown that the LCC-BR algorithm exhibits a lower decoding complexity and latency for high rate codes, which has been validated by the numerical results. Simulation results have shown the complexity and latency advantages of the proposed algorithms over the existing algorithms. The current work is limited in the interpolation multiplicity, trading the error-correction capability of each Chase decoding event for a lower basis construction complexity. Generalizing the proposed mechanism for larger multiplicities will be considered in future.

References

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

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

[3] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, "A method for solving key equation for decoding goppa codes," *Inf. Control*, vol. 27, no. 1, pp. 87–99, Jan. 1975.

[4] L. Welch and E. Berlekamp, "Error correction for algebraic block codes," U.S. Patent 4 633 470, Dec. 30, 1986.

[5] G. Forney, "Generalized minimum distance decoding," *IEEE Trans. Inf. Theory*, vol. IT-12, no. 2, pp. 125–131, Apr. 1966.

[6] D. Chase, "Class of algorithms for decoding block codes with channel measurement information," *IEEE Trans. Inf. Theory*, vol. IT-18, no. 1, pp. 170–182, Jan. 1972.

[7] V. Guruswami and M. Sudan, "Improved decoding of Reed–Solomon and algebraic-geometric codes," *IEEE Trans. Inf. Theory*, vol. 45, no. 1, pp. 1757–1767, Mar. 1999.

[8] R. Koetter and A. Vardy, "Algebraic soft-decision decoding of Reed–Solomon codes," *IEEE Trans. Inf. Theory*, vol. 49, no. 11, pp. 2809–2825, Nov. 2003.

[9] R. Kötter, "On algebraic decoding of algebraic-geometric and cyclic codes," Ph.D. dissertation, Dept. Elect. Eng., Univ. Linköping, Linköping, Sweden, 1996.

[10] H. O'Keeffe and P. Fitzpatrick, "Gröbner basis solutions of constrained interpolation problems," *Linear Algebra Appl.*, vol. 351, pp. 533–551, Aug. 2002.

[11] D. Lazard, "Ideal bases and primary decomposition: Case of two variables," *J. Symbolic Comput.*, vol. 1, no. 3, pp. 261–270, Sep. 1985.

[12] K. Lee and M. O'Sullivan, "List decoding of Reed–Solomon codes from a Gröbner basis perspective," *J. Symb. Comput.*, vol. 43, no. 9, pp. 645–658, Sep. 2008.

[13] K. Lee and M. O'Sullivan, "An interpolation algorithm using Gröbner bases for soft-decision decoding of Reed–Solomon codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Seattle, WA, USA, Jul. 2006, pp. 2032–2036.

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

[15] M. Alekhnovich, "Linear diophantine equations over polynomials and soft decoding of Reed–Solomon codes," *IEEE Trans. Inf. Theory*, vol. 51, no. 7, pp. 2257–2265, Jul. 2005.

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

[17] F. Gustavson and D. Yun, "Fast algorithms for rational Hermite approximation and solution of toeplitz systems," *IEEE Trans. Circuits Syst.*, vol. CS-26, no. 9, pp. 750–755, Sep. 1979.

[18] B. Beckermann, "A reliable method for computing M-Padé approximants on arbitrary staircases," *J. Comput. Appl. Math.*, vol. 40, no. 1, pp. 19–42, Jun. 1992.

[19] B. Beckermann and G. Labahn, "Fraction-free computation of matrix rational interpolants and matrix GCDs," *J. Matrix Anal. Appl.*, vol. 22, no. 1, pp. 114–144, May 2000.

[20] V. Olshevsky and M. A. Shokrollahi, "A displacement approach to efficient decoding of algebraic-geometric codes," in *Proc. ACM Symp. Theory Comput. (STOC)*, Atlanta, GA, USA, May 1999, pp. 235–244.

[21] A. Zeh, C. Gentner, and D. Augot, "An interpolation procedure for list decoding Reed–Solomon codes based on generalized key equations," *IEEE Trans. Inf. Theory*, vol. 57, no. 9, pp. 5946–5959, Sep. 2011.

[22] M. F. I. Chowdhury, C.-P. Jeannerod, V. Neiger, E. Schost, and G. Villard, "Faster algorithms for multivariate interpolation with multiplicities and simultaneous polynomial approximations," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2370–2387, May 2015.

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

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

[25] J. Ma and A. Vardy, "A complexity reducing transformation for the Lee-O'Sullivan interpolation algorithm," in *Proc. IEEE Int. Symp. Inf. Theory*, Nice, France, Jun. 2007, pp. 1986–1990.

[26] J. Xing, L. Chen, and M. Bossert, "Low-complexity Kotter-Vardy decoding of Reed–Solomon codes using module minimization," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Shanghai, China, May 2019, pp. 1–6.

[27] 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.

[28] J. Nielsen and A. Zeh, "Multi-trial Guruswami–Sudan decoding for generalised Reed–Solomon codes," *Des., Codes Cryptogr.*, pp. 507–527, Feb. 2014.

[29] J. Xing, L. Chen, and M. Bossert, "Progressive algebraic soft-decision decoding of Reed–Solomon codes using module minimization," *IEEE Trans. Commun.*, vol. 67, no. 11, pp. 7379–7391, Nov. 2019.

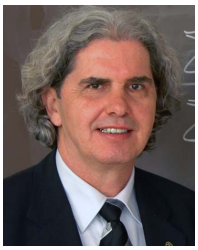
[30] J. Bellorado and A. Kavčić, "Low-complexity soft-decoding algorithms for Reed–Solomon codes—Part I: An algebraic soft-in hard-out chase decoder," *IEEE Trans. Inf. Theory*, vol. 56, no. 3, pp. 945–959, Mar. 2010.

[31] X. Zhang and Y. Zheng, "Generalized backward interpolation for algebraic soft-decision decoding of Reed–Solomon codes," *IEEE Trans. Commun.*, vol. 61, no. 1, pp. 13–23, Jan. 2013.

[32] J. Zhao, L. Chen, X. Ma, and M. Johnston, "Progressive algebraic chase decoding algorithms for Reed–Solomon codes," *IET Commun.*, vol. 10, no. 12, pp. 1416–1427, 2016.
[33] T. Kaneko, T. Nishijima, H. Inazumi, and S. Hirasawa, "An efficient maximum-likelihood-decoding algorithm for linear block codes with algebraic decoder," *IEEE Trans. Inf. Theory*, vol. 40, no. 2, pp. 320–327, Mar. 1994.
[34] L. Chen and M. Bossert, "Algebraic chase decoding of Reed–Solomon codes using module minimisation," in *Proc. Int. Symp. Inf. Theory App. (ISITA)*, Monterey, CA, USA, Oct. 2016, pp. 305–309.
[35] H. Hasse, "Theorie der höheren Differentiale in einem algebraischen Funktionenkörper mit vollkommenem Konstantenkörper bei beliebiger Charakteristik," *J. Reine. Angewandte Math.*, vol. 175, pp. 50–54, 1936.
[36] R. M. Roth and G. Ruckenstein, "Efficient decoding of Reed–Solomon codes beyond half the minimum distance," *IEEE Trans. Inf. Theory*, vol. 46, no. 1, pp. 246–257, Jan. 2000.
[37] G. Forney, "On decoding BCH codes," *IEEE Trans. Inf. Theory*, vol. IT-11, no. 4, pp. 549–557, Oct. 1965.
[38] R. P. Brent, F. G. Gustavson, and D. Y. Y. Yun, "Fast solution of toeplitz systems of equations and computation of Padé approximants," *J. Algorithms*, vol. 1, no. 3, pp. 259–295, Sep. 1980.
[39] M. Bossert, *Channel Coding for Telecommunications*. Hoboken, NJ, USA: Wiley, 1999.
[40] J. Xing, L. Chen, and M. Bossert, "Progressive module minimization for re-encoding transformed soft decoding of RS codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 1547–1551.

Jiongyue Xing (Graduate Student Member, IEEE) received the B.Sc. degree in communication engineering and the Ph.D. degree in information and communication engineering from Sun Yat-sen University, Guangzhou, China, in 2015 and 2020, respectively. From December 2018 to December 2019, he was a Visiting Ph.D. Student with the Institute of Communication Engineering, Ulm University, Germany. His research interests include channel coding and data communications.

Li Chen (Senior Member, IEEE) received the B.Sc. degree in applied physics from Jinan University, China, in 2003, and the M.Sc. degree in communications and signal processing and the Ph.D. degree in communications engineering from Newcastle University, U.K., in 2004 and 2008, respectively. From 2007 to 2010, he was a Research Associate with Newcastle University. In 2010, he returned to China as a Lecturer of the School of Information Science and Technology, Sun Yat-sen University, Guangzhou. From 2011 to 2012, he was a Visiting Researcher with the Institute of Network Coding, The Chinese University of Hong Kong. From 2011 and 2016, he was an Associate Professor and a Professor of university. Since 2013, he has been the Associate Head of the Department of Electronic and Communication Engineering (ECE). From July 2015 to October 2015, he was a Visitor of the Institute of Communications Engineering, Ulm University, Germany. From October 2015 to June 2016, he was a Visiting Associate Professor with the Department of Electrical Engineering, University of Notre Dame, USA. From 2017 to 2020, he was the Deputy Dean of the School of Electronics and Communication Engineering. His research interests include information theory, error-correction codes, and data communications. He likes reading and photography. He is a Senior Member of the Chinese Institute of Electronics (CIE). He is a member of the IEEE Information Theory Society Board of Governors Conference Committee, the Chair of the IEEE Information Theory Society Guangzhou Chapter, and a Committee Member of the CIE Information Theory Society. He is currently serving as an Associate Editor for IEEE TRANSACTIONS ON COMMUNICATIONS. He has been involved in organizing several international conferences, including the 2018 IEEE Information Theory Workshop (ITW) at Guangzhou, for which he was the General Co-Chair.



Martin Bossert (Fellow, IEEE) received the Dipl.Ing. degree in electrical engineering from the Karlsruhe Institute of Technology, Germany, in 1981, and the Ph.D. degree from the Technical University of Darmstadt, Germany, in 1987. After a one-year DFG scholarship at Linköping University, Sweden, he joined the AEG Mobile Communication, where he was involved in the specification and development of the GSM systems. Since 1993, he has been a Professor with Ulm University, Germany, where he is currently the Director of the Institute of Communications Engineering. He is the author of several textbooks and coauthor of more than 200 articles. His research interest includes reliable and secure data transmission. His main focus is on decoding of algebraic codes with reliability information and coded modulation. He has been a member of the IEEE Information Theory Society Board of Governors from 2010 to 2012 and has been appointed as a member of the German National Academy of Sciences Leopoldina in 2013. Among other awards and honors, he received the Vodafone Innovationspreis in 2007.