Low-Complexity Chase Decoding of Reed-Solomon Codes through Basis Reduction

Jiongyue Xing †, Li Chen †, Martin Bossert ‡
† School of Electronics and Information Technology, 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*—This paper proposes the low-complexity Chase (LCC) decoding using basis reduction (BR) interpolation for Reed-Solomon (RS) codes, namely the LCC-BR algorithm. With received soft information, a number of decoding test-vectors are formulated. The LCC-BR algorithm first constructs a common basis which will be utilized by the following individual basis constructions of all test-vectors. This eliminates the redundant computation in BR interpolation, resulting in a low decoding complexity. Moreover, the LCC-BR algorithm can decode each test-vector in parallel, lowering the decoding latency. This paper further proposes the progressive LCC-BR (PLCC-BR) algorithm that decodes the test-vectors sequentially and terminates once the intended message is found. This progressive decoding is realized without additional memory cost. Simulation results show the complexity and latency advantages of the proposed algorithms over the other benchmark algorithms.**

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

I. Introduction

In data communications and storage systems, Reed-Solomon (RS) codes are among the most popular error-correction codes. Currently, they are decoded by the efficient Berlekamp-Massey (BM) algorithm [1]. The algebraic Guruswami-Sudan (GS) [2] and Koetter-Vardy (KV) [3] algorithms yield a better performance but with a higher complexity. This is due to the construction of the interpolation polynomial, which is usually realized by Koetter's algorithm [4]. It can be facilitated by the re-encoding transform [5] and the progressive interpolation [6]. By identifying η unreliable received symbols, the low-complexity Chase (LCC) decoding algorithm [7] formulates 2^η test-vectors. It reduces the decoding complexity by exploiting the similarity among all test-vectors, eliminating the redundant computation in determining the interpolation polynomial for each of them. Several variants of the LCC algorithm have been proposed, including the hardware friendly backward-forward LCC (BF-LCC) algorithm [8] and the progressive LCC (PLCC) algorithm [9].

The interpolation polynomial can also be determined by the basis reduction (BR)[1] technique which is based on the concept of Gröbner basis of a module [10]. It first constructs a basis of a module satisfying all interpolation constraints, and then reduces it into a Gröbner basis that contains the intended polynomial. For practical codes, the basis reduction can be efficiently realized by the Mulders-Storjohann (MS) algorithm [11]. The module minimization (MM)[2] has been used in the re-encoding transformed (ReT) KV decoding and the progressive algebraic soft decoding (PASD), namely the ReT-KV-MM [12] and the PASD-MM [13] algorithms, respectively.

Recently, the MM based algebraic Chase decoding (ACD-MM) algorithm has been proposed in [14]. Unlike the LCC algorithm [7], it can perform parallel decoding for each test-vector, lowering the decoding latency. However, the ACD-MM algorithm does not fully utilize the similarity among the test-vectors, resulting in redundant decoding computation. To further optimize the decoding complexity, this paper proposes the BR based LCC (LCC-BR) algorithm for realizing low-complexity and low-latency decoding of RS codes. It will be shown that the BR interpolation of all test-vectors can be partitioned into the common basis construction and the individual basis construction. The common basis construction 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 all decoding trials. Furthermore, this paper proposes the progressive LCC-BR (PLCC-BR) algorithm, which can remove the memory requirement of the original PLCC algorithm [9]. Our simulation results will demonstrate the complexity and latency advantages of the proposed algorithms over several existing ones.

II. Background Knowledge

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 and dimension k, codeword $\underline{c} = (c_0, c_1, \ldots, c_{n-1}) \in \mathbb{F}_q^n$ can be generated by

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

where

$$f(x) = f_0 + f_1 x + \cdots + f_{k-1}x^{k-1} \in \mathbb{F}_q[x]$$

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

[1]In our earlier publications, we named this interpolation technique module minimization (MM). However, in the course of our research, we realized that basis reduction (BR) is a more appropriate name. The basis of a module is reduced instead of the module itself.

[2]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.

B. The BR Based GS Algorithm

Let $\underline{\omega} = (\omega_0, \omega_1, \dots, \omega_{n-1}) \in \mathbb{F}_q^n$ denote the hard-decision received word. Given $Q(x, y) = \sum_{a,b} Q_{ab} x^a y^b \in \mathbb{F}_q[x, y]$ and a nonnegative integer pair (κ, ι), the (κ, ι)-Hasse derivative evaluation at one point (α_j, ω_j) is defined as

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

If $D_{\kappa,\iota}^{(\alpha_j,\omega_j)}(Q(x, y)) = 0, \forall\kappa + \iota < m$, $Q(x, y)$ interpolates (α_j, ω_j) with a multiplicity m. By defining the n interpolation points $(\alpha_0, \omega_0), (\alpha_1, \omega_1), \dots, (\alpha_{n-1}, \omega_{n-1})$, the GS algorithm first constructs an interpolation polynomial $Q(x, y) \in \mathbb{F}_q[x, y]$ that passes through each of them with a multiplicity m. Let l denote the y-degree of $Q(x, y)$, i.e., $l = \deg_y Q$. Note that $m \leq l$. After generating Q, message polynomial $f(x)$ can be recovered by finding its y-roots, i.e., $Q(x, f(x)) = 0$ [2].

For a bivariate monomial $x^a y^b$, its (μ, ν)-weighted degree is $\deg_{\mu,\nu} x^a y^b = \mu a + \nu b$. Given $Q(x, y) = \sum_{a,b} Q_{ab} x^a y^b$, its coefficients can be organized under the (μ, ν)-reverse lexicographic (revlex) order, which 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 $b_1 < b_2$. Further let $x^{a'}y^{b'}$ be the leading monomial (LM) of Q with $Q_{a'b'} \neq 0$, 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 $\mathrm{LM}(Q_1) = x^{a'_1}y^{b'_1}$ and $\mathrm{LM}(Q_2) = x^{a'_2}y^{b'_2}$, respectively, $Q_1 < Q_2$ if $\mathrm{LM}(Q_1) < \mathrm{LM}(Q_2)$.

The interpolation polynomial Q can be constructed by the BR technique [10]. It first constructs a basis of a module, which will then be reduced into a Gröbner basis that contains Q. Module $\mathcal{M}$ is defined as follows.

Definition I. 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.

In order to construct a module basis, the following two module seeds are needed,

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

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. Note that $R(\alpha_j) = \omega_j, \forall j$. Based on Definition I, $\mathcal{M}$ can be generated as an $\mathbb{F}_q[x]$-module by the following $l + 1$ polynomials

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

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

They form a basis of a module, denoted as $\mathcal{B}$.

Lemma 1 [10]. 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 reduction is to perform $\mathbb{F}_q[x]$-linear combinations on $P_t(x, y)$ until the y-degree of all $\mathrm{LM}(P_t)$ are different, resulting in a Gröbner basis of $\mathcal{B}$. The minimum candidate is chosen as the interpolation polynomial $Q(x, y)$. In this work, we apply the MS algorithm [11] for the basis reduction.

III. The LCC-BR algorithm

This section proposes the LCC-BR algorithm, in which we set $m = l = 1$. It starts with formulating the test-vectors.

A. Test-vectors Formulation

Assume codeword $\underline{c}$ is transmitted through a memoryless channel and $\underline{\mathrm{r}} = (\mathrm{r}_0, \mathrm{r}_1, \dots, \mathrm{r}_{n-1}) \in \mathbb{R}^n$ is the received symbol vector. By assuming $\Pr[c_j = \sigma_i] = \frac{1}{q}$, an *a posteriori* probability matrix $\mathbf{\Pi} \in \mathbb{R}^{q\times n}$ with entries $\pi_{ij} = \Pr[c_j = \sigma_i \mid \mathrm{r}_j]$ can be observed, where $0 \leq i \leq q - 1$ and $0 \leq j \leq n - 1$. For each r_j, the two most likely decisions are $r_j^{\mathrm{I}} = \sigma_{i_j^{\mathrm{I}}}$ and $r_j^{\mathrm{II}} = \sigma_{i_j^{\mathrm{II}}}$, where $i_j^{\mathrm{I}} = \arg\max_i\{\pi_{ij}\}$ and $i_j^{\mathrm{II}} = \arg\max_{i,i\neq i_j^{\mathrm{I}}}\{\pi_{ij}\}$, respectively. Hence, the symbol-wise reliability measure of r_j can be defined as

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

where $\gamma_j \in (1, \infty)$. The decision on r_j is more reliable if γ_j is larger, and vice versa. A new symbol index sequence $j_0, j_1, \dots, j_{n-1}$ is yielded by sorting γ_j in an descenting order. It indicates $\gamma_{j_0} \geq \gamma_{j_1} \geq \cdots \geq \gamma_{j_{n-1}}$. By identifying η unreliable symbols, we define the reliable index set $\Theta = \{j_0, j_1, \dots, j_{n-\eta-1}\}$ and the unreliable index set $\Theta^{\mathrm{c}} = \{j_{n-\eta}, j_{n-\eta+1}, \dots, j_{n-1}\}$, respectively. Consequently, all test-vectors can be written as

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

where $u = 1, 2, \dots, 2^\eta$, among which $r_j^{(u)} = r_j^{\mathrm{I}}$ for $j \in \Theta$ and $r_j^{(u)} = r_j^{\mathrm{I}}$ or r_j^{II} for $j \in \Theta^{\mathrm{c}}$.

B. Re-encoding Transform

The LCC-BR algorithm is facilitated by the re-encoding transform [5]. Let $\eta \leq n - k$ so that all test-vectors would share at least k common symbols $r_{j_0}^{\mathrm{I}}, r_{j_1}^{\mathrm{I}}, \dots, r_{j_{k-1}}^{\mathrm{I}}$. Let $\Psi = \{j_0, j_1, \dots, j_{k-1}\}$ denote the index set of the k most reliable symbols, and $\Psi^{\mathrm{c}} = \{j_k, j_{k+1}, \dots, j_{n-1}\}$. A re-encoding codeword $\underline{h} = (h_0, h_1, \dots, h_{n-1})$ can be generated by setting $h_j = r_j^{\mathrm{I}}, \forall j \in \Psi$ and determining the remaining $n - k$ symbols by Forney's algorithm [15]. All test-vectors $\underline{r}_u$ are then transformed by

$$\underline{r}_u \mapsto \underline{z}_u : z_j^{(u)} = r_j^{(u)} - h_j, \forall j. \tag{6}$$

Consequently, the transformed test-vectors become

$$\underline{z}_u = (0, 0, \dots, 0, z_{j_k}^{(u)}, \dots, z_{j_{n-1}}^{(u)}). \tag{7}$$

C. Common Basis Construction

From (5), we observe that the 2^η test-vectors share common interpolation points $(\alpha_j, r_j^{\mathrm{I}}), \forall j \in \Theta$. 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 $\Psi' = \Psi^{\mathrm{c}} \setminus \Theta^{\mathrm{c}} = \{j_k, j_{k+1}, \ldots, j_{n-\eta-1}\}$. We define a common test-vector $\underline{z}_0 = (z_0^{(0)}, z_1^{(0)}, \ldots, z_{n-1}^{(0)})$, where $z_j^{(0)} = z_j^{\mathrm{I}}, \forall j \in \Psi'$ and $z_j^{(0)} = 0, \forall j \in \Psi \cup \Theta^{\mathrm{c}}$. For all test-vectors $\underline{z}_u$, the polynomial (2) is redefined as

$$R_u(x) = \sum_{j=0}^{n-1} z_j^{(u)} T_j(x). \tag{8}$$

Since $z_j^{(u)} = 0, \forall j \in \Psi$, $V(x) = \prod_{j \in \Psi}(x - \alpha_j)$ becomes the GCD for both $G(x)$ of (1) and the above $R_u(x)$. Therefore, two new module seeds are defined as

$$\tilde{G}(x) = \frac{G(x)}{V(x)} = \prod_{j \in \Psi^{\mathrm{c}}} (x - \alpha_j) \tag{9}$$

and

$$\tilde{R}_u(x) = \frac{R_u(x)}{V(x)} = \sum_{j \in \Psi^{\mathrm{c}}} z_j^{(u)} \tilde{T}_j(x), \tag{10}$$

where

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

With $m = l = 1$, the module generators (3) and (4) for the common test-vector $\underline{z}_0$ can be simplified into

$$\tilde{P}_{0,0}(x, y) = \tilde{G}(x), \tag{11}$$

$$\tilde{P}_{0,1}(x, y) = y - \tilde{R}_0(x), \tag{12}$$

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 points $(\alpha_j, z_j^{(0)}), \forall j \in \Psi'$ with a multiplicity of one, constructing the common basis $\tilde{\mathcal{B}}_0$. The MS algorithm [11] that performs $\mathbb{F}_q[x]$-linear combinations will reduce $\tilde{\mathcal{B}}_0$ into a Gröbner basis $\tilde{\mathcal{B}}_0'$ which contains polynomials $\tilde{P}_{0,0}'(x, y)$ and $\tilde{P}_{0,1}'(x, y)$. Based on Lemma 1, they can be written as

$$\tilde{P}_{0,t}'(x, y) = p_{t0}(x)\tilde{G}(x) + p_{t1}(x)(y - \tilde{R}_0(x)), \tag{13}$$

where $p_{t0}(x), p_{t1}(x) \in \mathbb{F}_q[x]$ and $t = 0, 1$. They will be utilized by the following 2^η individual basis constructions.

D. 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 (10) 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 (13), for each test-vector $\underline{z}_u$, we have

$$\begin{aligned} \tilde{P}_{u,t}(x, y) &= p_{t0}(x)\tilde{G}(x) + p_{t1}(x)(y - \tilde{R}_u(x)) \\ &= p_{t0}(x)\tilde{G}(x) + p_{t1}(x)(y - \tilde{R}_0(x) - \tilde{\Upsilon}_u(x)) \\ &= \tilde{P}_{0,t}'(x, y) - p_{t1}(x)\tilde{\Upsilon}_u(x). \end{aligned} \tag{14}$$

Polynomials $\tilde{P}_{u,0}(x, y)$ and $\tilde{P}_{u,1}(x, y)$ form the basis $\tilde{\mathcal{B}}_u$. Since all $\tilde{\mathcal{B}}_u$ are constructed using the previously reduced common basis $\tilde{\mathcal{B}}_0'$, the redundant computation in decoding all test-vectors can be eliminated. Afterwards, the MS algorithm will reduce each basis $\tilde{\mathcal{B}}_u$ into its Gröbner basis $\tilde{\mathcal{B}}_u'$ which contains polynomials $\tilde{P}_{u,0}'(x, y)$ and $\tilde{P}_{u,1}'(x, y)$, and $\tilde{Q}_u(x, y) = \min\{\tilde{P}_{u,0}'(x, y), \tilde{P}_{u,1}'(x, y)\}$. Note that with the re-encoding transform, polynomials are organized by the $(1, -1)$-revlex order [5]. 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

$$Q_u(x, y) = V(x)\tilde{Q}_u^{(0)}(x) + \tilde{Q}_u^{(1)}(x)y. \tag{15}$$

Now $Q_u(x, y)$ interpolates $(\alpha_j, z_j^{(u)}), \forall j$ with a multiplicity of one. If the decoding estimation $\tilde{f}_u(x)$ satisfies $Q_u(x, \tilde{f}_u(x)) = 0$, $\tilde{f}_u(x)$ can be determined by

$$\tilde{f}_u(x) = -\frac{V(x)\tilde{Q}_u^{(0)}(x)}{\tilde{Q}_u^{(1)}(x)}, \tag{16}$$

and the estimated codeword is $\underline{\hat{c}}_u = (\hat{c}_0^{(u)}, \hat{c}_1^{(u)}, \ldots, \hat{c}_{n-1}^{(u)})$, where $\hat{c}_j^{(u)} = \tilde{f}_u(\alpha_j) + h_j, \forall j$. Its corresponding message polynomial is denoted as $\hat{f}_u(x)$. If $\tilde{Q}_u^{(1)}(x) \nmid V(x)\tilde{Q}_u^{(0)}(x)$, the decoding of test-vector $\underline{z}_u$ fails. After decoding all 2^η test-vectors, the message $\hat{f}_u(x)$ that yields the most likely codeword among all candidates will be chosen as the output.

The LCC-BR algorithm is summarized in Algorithm 1.

Algorithm 1 The LCC-BR Algorithm

```
Input: $\mathbf{\Pi}, \eta$;
Output: $\hat{f}_u(x)$;
1: Formulate $2^\eta$ test-vectors $\underline{r}_u$ as in (5);
2: Perform re-encoding transform to yield $\underline{z}_u$ as in (6);
3: Construct $\tilde{\mathcal{B}}_0$ by (11) (12) and reduce it into $\tilde{\mathcal{B}}_0'$;
4: For each transformed test-vector $\underline{z}_u$ do
5:     Construct $\tilde{\mathcal{B}}_u$ by (14);
6:     Perform the MS algorithm to reduce $\tilde{\mathcal{B}}_u$ into $\tilde{\mathcal{B}}_u'$;
7:     Determine $Q_u(x, y)$ as in (15);
8:     Decode $\tilde{f}_u(x)$ as in (16) and generate $\hat{f}_u(x)$;
9: End for
```

Remark 1. Unlike the LCC algorithm [7], the LCC-BR algorithm can decode all test-vector in parallel, yielding a low decoding latency. It also advances from the existing ACD-MM algorithm [14] by eliminating the redundant computation on basis construction and reduction for all 2^η test-vectors.

IV. The Progressive Variant

This sections proposes the PLCC-BR algorithm, in which the test-vectors are decoded sequentially. It will first order the test-vectors such that the one with a higher potential of yielding the intended message will be decoded earlier.

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 $\Omega_u = \prod_{j=0}^{n-1} \pi_{i_j^{(u)} j}$, where $i_j^{(u)} = \mathrm{index}\{\sigma_i \mid \sigma_i = r_j^{(u)}\}$ [9]. The test-vector with a larger Ω_u is considered 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 simplified into

$$\tilde{\Omega}_u = \prod_{j \in \Theta^{\mathrm{c}}} \pi_{i_j^{(u)} j}. \tag{17}$$

By sorting $\tilde{\Omega}_u$ in a descending order such that $\tilde{\Omega}_{u_1} > \tilde{\Omega}_{u_2} > \cdots > \tilde{\Omega}_{u_{2^\eta}}$, the progressive decoder will first decode test-vectors $\underline{r}_{u_1}$, then decode $\underline{r}_{u_2}$ and etc. It will terminate when a codeword that satisfies the maximum-likelihood (ML) criterion [16] is found. Note that $\underline{r}_{u_1} = \underline{\omega}$.

B. The PLCC-BR Algorithm

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

$$\Lambda_{u_\tau} = \{j \mid r_j^{(u_\tau)} \neq r_j^{(u_{\tau+1})}, j \in \Theta^c\}. \tag{18}$$

The PLCC-BR algorithm is described as follows. At the beginning, a basis $\tilde{\mathcal{B}}_{u_1}$ for test-vector $\underline{z}_{u_1}$ is constructed by

$$\tilde{P}_{u_1,0}(x,y) = \tilde{G}(x), \tag{19}$$

$$\tilde{P}_{u_1,1}(x,y) = y - \tilde{R}_{u_1}(x). \tag{20}$$

By performing Steps 6 – 8 of Algorithm 1, we can determine the estimated codeword $\hat{\underline{c}}_{u_1}$ and its corresponding message $\hat{f}_{u_1}(x)$. If $\hat{\underline{c}}_{u_1}$ satisfies the ML criterion [16], the decoding terminates and outputs $\hat{f}_{u_1}(x)$. Otherwise, the decoding continues to decode test-vector $\underline{z}_{u_2}$. Based on Lemma 1, polynomials $\tilde{P}'_{u_1,0}(x,y)$ and $\tilde{P}'_{u_1,1}(x,y)$ of basis $\tilde{\mathcal{B}}'_{u_1}$ can be written as

$$\tilde{P}'_{u_1,t}(x,y) = p_{u_1,t0}(x)\tilde{G}(x) + p_{u_1,t1}(x)(y - \tilde{R}_{u_1}(x)),$$

where $p_{u_1,t0}(x), p_{u_1,t1}(x) \in \mathbb{F}_q[x]$ and $t = 0, 1$. The difference between the test-vectors $\underline{z}_{u_1}$ and $\underline{z}_{u_2}$ is utilized to formulate the polynomial $\tilde{R}_{u_2}(x)$, which can be obtained by

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

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 test-vector $\underline{z}_{u_2}$ can be straightforwardly constructed by

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

Again, the estimated codeword $\hat{\underline{c}}_{u_2}$ is decoded to see if it satisfies the ML criterion.

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

$$\tilde{P}_{u_\tau,t}(x,y) = \tilde{P}'_{u_{\tau-1},t}(x,y) - p_{u_{\tau-1},t1}(x)W_{u_{\tau-1}}(x), \tag{21}$$

where $t = 0, 1$ and

$$\begin{aligned}W_{u_{\tau-1}}(x) &= \sum_{j\in\Lambda_{u_{\tau-1}}}(z_j^{(u_\tau)} - z_j^{(u_{\tau-1})})\tilde{T}_j(x) \\ &= \sum_{j\in\Lambda_{u_{\tau-1}}}(r_j^{(u_\tau)} - r_j^{(u_{\tau-1})})\tilde{T}_j(x).\end{aligned} \tag{22}$$

Note that $\tilde{P}'_{u_{\tau-1},t}(x,y)$ and $p_{u_{\tau-1},t1}(x)$ are obtained from the decoding of test-vector $\underline{z}_{u_{\tau-1}}$. After reducing basis $\tilde{\mathcal{B}}_{u_\tau}$ into its Gröbner basis $\tilde{\mathcal{B}}'_{u_\tau}$, the interpolation polynomial $Q_{u_\tau}(x,y)$ is determined by (15). If it produces a codeword $\hat{\underline{c}}_{u_\tau}$ that satisfies the ML criterion, the decoding terminates and outputs $\hat{f}_{u_\tau}(x)$. Otherwise, the decoding continues to decode test-vector $\underline{z}_{u_{\tau+1}}$. If the decoding of all 2^η test-vectors cannot produce an ML codeword, the PLCC-BR algorithm terminates with a failure.

Remark 2. Eqs. (20) and (22) reveal that the re-encoding transform only needs to be performed once, that is for $\underline{r}_{u_1}$.

Remark 3. It can be observed from (21) that the PLCC-BR algorithm does not need to memorize the intermediate decoding information, while the original PLCC algorithm exhibits the additional memory cost of $O(2^\eta(n-k))$ [9].

V. Simulation Results

This section shows the decoding complexity, latency and performance of the proposed algorithms. They are measured as the number of finite field multiplications, the running time required to decode a codeword and the frame error rate (FER), respectively. Our results were obtained over the additive white Gaussian noise (AWGN) channel using BPSK.

A. Decoding Complexity

TABLE I
Complexity comparison in decoding the (63, 47) RS code

η	LCC [7]	BF-LCC [8]	ACD-MM [14]	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

Table I compares complexity of the LCC-BR algorithm with several existing Chase decoding algorithms in decoding the (63, 47) RS code. Note that they employ the same test-vectors formulation. Both the LCC [7] and the BF-LCC [8] algorithms employ Koetter's interpolation. Table I shows that the LCC-BR algorithm is less complex than the BF-LCC algorithm. Meanwhile, by eliminating the redundant BR computation, the proposed LCC-BR algorithm yields a lower complexity than the ACD-MM algorithm. However, it remains slightly more complex than the LCC algorithm which performs the interpolation of all test-vectors in a binary tree growing fashion, granting it a low complexity feature.

TABLE II
Average complexity of the PLCC-BR and the PASD-MM algorithms in decoding the (63, 47) RS code

SNR	PLCC-BR			PASD-MM [13]
(dB)	$\eta = 2$	$\eta = 4$	$\eta = 6$	$l = 4$
3.0	1.76×10^4	5.67×10^4	2.08×10^5	5.12×10^5
3.5	1.69×10^4	5.46×10^4	2.01×10^5	4.64×10^5
4.0	1.53×10^4	4.83×10^4	1.73×10^5	3.48×10^5
4.5	1.14×10^4	2.72×10^4	1.07×10^5	1.58×10^5
5.0	9.98×10^3	1.39×10^4	4.38×10^4	6.55×10^4
5.5	8.73×10^3	9.62×10^3	2.07×10^4	3.09×10^4
6.0	8.32×10^3	8.52×10^3	8.82×10^3	1.93×10^4
6.5	8.22×10^3	8.22×10^3	8.22×10^3	1.87×10^4
7.0	8.18×10^3	8.18×10^3	8.18×10^3	1.85×10^4

To show the effectiveness of the PLCC-BR algorithm, we measure its average complexity over multiple decoding events at a certain signal-to-noise ratio (SNR). Table II shows the average complexity in decoding the (63, 47) RS code. It shows that complexity advantage of the progressive decoding becomes more obvious as the SNR increases, where the decoding can terminate earlier. For this code, complexity of the BM algorithm is 2.42×10^3. When SNR $\geq$ 6 dB, the progressive decoding complexity can converge to the minimum level that has the same magnitude as the BM complexity.

B. Decoding Latency

TABLE III
DECODING LATENCY (MS) COMPARISON BETWEEN THE LCC AND THE LCC-BR ALGORITHMS

η	LCC		LCC-BR	
	(255, 127) RS	(255, 239) RS	(255, 127) RS	(255, 239) RS
2	6.687	7.921	34.061	1.687
4	19.029	27.769	33.684	1.632
6	70.038	104.337	33.471	1.604

Table III shows the latency (in ms) in decoding two RS codes defined over $\mathbb{F}_{256}$. These results were obtained by implementing the algorithms in C and running the program on Intel core i5-4260U CPU and macOS operating system. Since the LCC-BR algorithm can perform its individual basis construction and root-finding in parallel, the re-encoding transform and common basis construction will dominate the running time. Therefore, the decoding latency does not vary significantly by changing η and it is defined by that of decoding a single test-vector. For the LCC algorithm, running time increases as η enlarges due to exponentially expanded interpolation. 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, revealing its effectiveness for high rate codes [13]. Compared with the LCC algorithm, the proposal shows its advantage on the decoding latency for large η and high rate codes, which will be more practically welcomed.

C. Decoding Performance

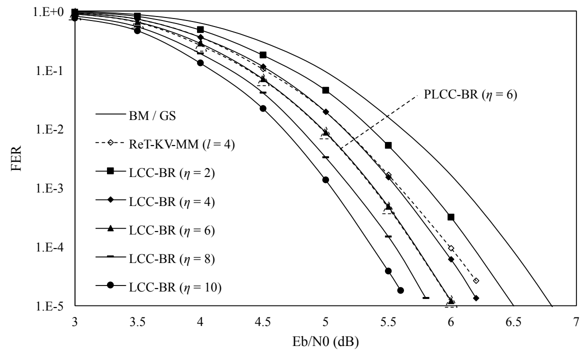

Fig. 1. Performance of the (63, 47) RS code.

Fig. 1 shows performance of the (63, 47) RS code. Note that the four Chase decoding algorithms shown in Table I yield the same performance since they decode the same test-vectors. As η increases, the LCC-BR performance improves since more test-vectors are decoded. When $\eta = 10$, the LCC-BR algorithm outperforms the BM algorithm with 1.1 dB gains at the FER of 10^{-4}. It can be seen that the PLCC-BR algorithm maintains the LCC-BR decoding performance, e.g., $\eta = 6$. Revisiting Tables I and II, the PLCC-BR algorithm has lower complexity than the LCC-BR algorithm over the whole SNR regions. Fig. 1 also shows that the ReT-KV-MM ($l = 4$) [12] performs similarly as the LCC-BR ($\eta = 4$), and its complexity is 4.98×10^5. Tables I and II show that the LCC-BR and the PLCC-BR algorithms exhibit a lower complexity than the ReT-KV-MM and the PASD-MM algorithms, respectively.

VI. CONCLUSION

This paper has proposed the BR interpolation based LCC algorithm for RS codes. It first constructs a common basis which is shared by the parallel decoding of the formulated test-vectors, removing the redundant computation in the BR interpolation and achieving a low decoding latency. The PLCC-BR algorithm has been further proposed to adjust the decoding computation to the channel quality. By exploiting the difference between the adjacent test-vectors, this progressive decoding is realized without additional memory cost. Simulation results have shown that the proposed algorithms are favorable compared with the other ones.

ACKNOWLEDGEMENT

This work is sponsored by 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*. PhD thesis, Univ. Linköping, Linköping, Sweden, 1996.

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

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

[8] J. Zhu, X. Zhang, and Z. Wang, "Backward interpolation architecture for algebraic soft-decision Reed-Solomon decoding," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 17, no. 11, pp. 1602–1615, Nov. 2009.

[9] 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.

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

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

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

[13] 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.

[14] L. Chen and M. Bossert, "Algebraic Chase decoding of Reed-Solomon codes using module minimisation," in *Proc. Int. Symp. Inform. Theory App. (ISITA)*, Monterey, U.S.A, pp. 310–314, Oct. 2016.

[15] G. Forney, "On decoding BCH codes," *IEEE Trans. Inform. Theory*, vol. 11, no. 4, pp. 549–557, Oct. 1965.

[16] T. Kaneko *et al.*, "An efficient maximum-likelihood-decoding algorithm for linear block codes with algebraic decoder," *IEEE Trans. Inform. Theory*, vol. 40, no. 2, pp. 320–327, Mar. 1994.