Progressive Algebraic Soft Decoding of Reed-Solomon Codes Using Module Minimization

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

***Abstract*—The algebraic soft decoding (ASD) algorithm achieves advanced decoding performance for Reed-Solomon (RS) codes. However, its complexity remains high making it impractical. This is due to the interpolation. The progressive ASD (PASD) algorithm adjusts the decoding computation to the reliability of received information. Its interpolation generates the intended polynomial $Q(x, y)$ with a progressively enlarged y-degree, and terminates once the message is decoded. But this progressive decoding is realized at the cost of memorizing the intermediate decoding information. This paper proposes a new PASD algorithm, in which the progressive interpolation is realized by the module minimization (MM) technique. Polynomial $Q(x, y)$ can be found through the progressively enlarged images of submodule's basis without memorizing the intermediate decoding information. The MM interpolation also grants it a significantly lower complexity than the original PASD algorithm that uses Koetter's interpolation. Our simulation results will verify its advanced decoding performance and low-complexity feature.**

***Index Terms*—Algebraic soft decoding, module minimization, progressive decoding, Reed-Solomon codes**

I. Introduction

Reed-Solomon (RS) codes are widely employed in communication systems and storage devices. The Berlekamp-Massey (BM) decoding algorithm [1] can correct errors up to half of the code's minimum Hamming distance. The late Guruswami-Sudan (GS) algebraic decoding algorithm can correct errors beyond this limit [2]. Utilizing soft information, the Koetter-Vardy algebraic soft decoding (ASD) algorithm can further enhance the decoding performance [3]. However, complexity of the algebraic decoding algorithms are still orders of magnitude higher than the BM algorithm. This is due to the interpolation that finds the minimum polynomial $Q(x, y)$. Currently, it is realized by Koetter's iterative polynomial construction algorithm [4]. There exists several approaches to facilitate Koetter's interpolation, including the re-encoding transform [5], Wu's algorithm [6] and the progressive interpolation [7]. The latter results in the progressive ASD (PASD) algorithm. It can adjust the decoding computation to the reliability of received information, decoding the message with the smallest parameter, i.e., $\deg_y Q$.

It has been reported that the interpolation problem can be solved using the concept of module [8]. One can formulate a basis of module which contains bivariate polynomials that interpolate all the prescribed points with their multiplicity. Presenting the basis as a matrix over univariate polynomials, the Mulders-Storjohann (MS) algorithm [9] can further reduce it into the Gröbner basis. The interpolated polynomial $Q(x, y)$ is the minimum candidate of the basis. This interpolation technique is called the module minimization (MM). Some work including [8] and [10] have shown that using this technique, the ASD algorithm is less computationally expensive in comparison to the case that uses Koetter's interpolation. In this paper, it is named the ASD-MM algorithm.

This paper introduces a new PASD algorithm in which its progressive interpolation is realized by the MM technique, namely the PASD-MM algorithm. In the original PASD algorithm [7], the Gröbner basis is expanded according to the progressively enlarged $\deg_y Q$. However, during the expansion, the newly introduced polynomial needs to be updated using the intermediate decoding information. Hence, its low-complexity feature is exchanged by memorizing the intermediate decoding information. Utilizing the MM interpolation, this memory requirement can be removed. It also results in a remarkably low complexity for the progressive soft decoding. We will show the interpolation can be realized through the progressively enlarged images of submodule's basis. During the enlargement, the newly introduced polynomial can be directly generated from the enumerated interpolation points, without the need of memorizing the intermediate decoding information. Its decoding performance for the (255, 239) RS code will be shown. Our numerical results will also verify the proposal's low-complexity and channel dependent features.

II. Background Knowledge

This section presents the background knowledge of the paper, including RS codes and the PASD algorithm.

A. RS Codes

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

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

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

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

978-1-5386-4780-6/18/$31.00©2018 IEEE

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

B. The PASD Algorithm

Assume codeword $\underline{c} = (c_0, c_1, \ldots, c_{n-1})$ is transmitted through a memoryless channel and $\underline{\mathsf{r}} = (\mathsf{r}_0, \mathsf{r}_1, \ldots, \mathsf{r}_{n-1}) \in \mathbb{R}^n$ is the received symbol vector. A reliability matrix $\mathbf{\Pi}$ of size $q \times n$ can be obtained. Its entry $\pi_{ij} = \Pr[c_j = \sigma_i \mid \mathsf{r}_j]$ is the symbol wise *a posteriori* probability (APP) [1]. Matrix $\mathbf{\Pi}$ will be transformed into a multiplicity matrix $\mathbf{M}$ of the same size [3], whose entry m_{ij} ($m_{ij} \neq 0$) indicates the interpolation multiplicity for point (α_j, σ_i). Interpolation finds the minimum polynomial $Q(x, y)$ that interpolates all points (α_j, σ_i) with their multiplicity m_{ij}. Given $Q(x, y) = \sum_{a,b} Q_{ab} x^a y^b \in \mathbb{F}_q[x, y]$, its $(1, k-1)$-weighted degree is $\deg_{1,k-1} Q(x, y) = \max\{a + (k-1)b \mid Q_{ab} \neq 0\}$. Let $i_j = \text{index}\{\sigma_i \mid \sigma_i = c_j\}$, codeword score is $S_{\mathbf{M}}(\underline{c}) = \sum_{j=0}^{n-1} m_{i_j j}$. The following theorem gives a sufficient condition for decoding $f(x)$.

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

Polynomial $Q(x, y)$ is often constructed using Koetter's algorithm [4]. Parameterized by $l = \deg_y Q$, Koetter's interpolation starts with a set of $l + 1$ polynomials, i.e., $\mathcal{G}_l = \{1, y, \ldots, y^l\}$. They are iteratively updated, evolving into the Gröbner basis in which Q is the minimum candidate [2]. In contrast, the PASD algorithm functions with a progressively enlarged l. Beginning with $l = 1$, it finds the minimum interpolated polynomial Q_1 ($\deg_y Q_1 = 1$) from a computed polynomial set $\mathcal{G}_1$ that is initialized as $\{1, y\}$. If $Q_1(x, f(x)) = 0$, the message can be decoded [3] and the decoding terminates. Otherwise, $\mathcal{G}_1$ will be expanded by introducing a new polynomial y^2. The new polynomial needs to be updated so that it satisfies the interpolation constraints that have been satisfied by the existing polynomials of $\mathcal{G}_1$, forming an expanded polynomial set $\mathcal{G}_2$ and $|\mathcal{G}_2| = 3$. The minimum interpolated polynomial Q_2 ($\deg_y Q_2 = 2$) will be selected from a further computed set $\mathcal{G}_2$. Again, if $Q_2(x, f(x)) = 0$, the decoding terminates. Otherwise, the decoding continues by enlarging l progressively. It terminates either when the message is decoded or when l exceeds a predefined value. Consequently, the PASD algorithm decodes the message with the smallest parameter l. The above description shows that the newly introduced polynomial needs to be updated using the intermediate decoding information. The progressive decoding system needs to memorize this information [7].

III. The ASD-MM algorithm

This section introduces the ASD-MM algorithm, which is a prototype of the PASD-MM algorithm.

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

[2]Given $Q_1, Q_2 \in \mathbb{F}_q[x, y]$, and $\deg_{1,k-1} Q_1 = a_1 + (k-1)b_1$ and $\deg_{1,k-1} Q_2 = a_2 + (k-1)b_2$, it is claimed $Q_1 < Q_2$ if $\deg_{1,k-1} Q_1 < \deg_{1,k-1} Q_2$, or $\deg_{1,k-1} Q_1 = \deg_{1,k-1} Q_2$ and $b_1 < b_2$.

[3]The message polynomial $f(x)$ can be validated using the maximum likelihood criterion of [11].

A. Module Formulation

In order to determine the minimum interpolated polynomial $Q(x, y)$ where $\deg_y Q = l$, module $\mathcal{M}_l$ is needed.

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

Given matrix $\mathbf{M}$, let

$$\mathsf{m}_j = \sum_{i=0}^{q-1} m_{ij} \tag{3}$$

and

$$\mathsf{m} = \max\{\mathsf{m}_j, \forall j\}. \tag{4}$$

The $\mathbf{\Pi} \to \mathbf{M}$ transform terminates when $\mathsf{m} = l$ [3]. In order to formulate $\mathcal{M}_l$, the following point enumeration is needed. Let L_j denote an enumeration list that is drawn from column j of $\mathbf{M}$. It contains interpolation points (α_j, σ_i) with their multiplicity m_{ij} as

$$L_j = \{\underbrace{(\alpha_j, \sigma_i), \ldots, (\alpha_j, \sigma_i)}_{m_{ij}}, \forall i \text{ and } m_{ij} \neq 0\}. \tag{5}$$

Note that $|L_j| = \mathsf{m}_j$. Its balanced list L'_j is further created. Initialize $L'_j = \emptyset$. Move one of the most frequent elements from L_j to L'_j and repeat this process m_j times. The balanced list can be denoted as

$$L'_j = \{(\alpha_j, y_j^{(0)}), (\alpha_j, y_j^{(1)}), \ldots, (\alpha_j, y_j^{(\mathsf{m}_j - 1)})\}, \tag{6}$$

where $y_j^{(0)}, y_j^{(1)}, \ldots, y_j^{(\mathsf{m}_j - 1)} \in \mathbb{F}_q$ and they may not be distinct. Again, $|L'_j| = \mathsf{m}_j$. Finally, let $\mathsf{m}_j(s)$ denote the maximum multiplicity of the last $\mathsf{m}_j - s$ elements of L'_j as

$$\mathsf{m}_j(s) = \max\{\text{multi.}((\alpha_j, y_j^{(\varepsilon)})) \mid \varepsilon = s, s+1, \ldots, \mathsf{m}_j - 1\}. \tag{7}$$

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

The module $\mathcal{M}_l$ can now be generated. First, let

$$F_\varepsilon(x) = \sum_{j=0}^{n-1} y_j^{(\varepsilon)} \Phi_j(x), \tag{8}$$

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

$$P_t(x, y) = \prod_{j=0}^{n-1} (x - \alpha_j)^{\mathsf{m}_j(t)} \prod_{\varepsilon=0}^{t-1} (y - F_\varepsilon(x)), \tag{9}$$

where $t = 0, 1, \ldots, l$. Note $\deg_y P_t(x, y) = t \leq l$. The above equation defines the $l+1$ generators for $\mathcal{M}_l$. Since any element of $\mathcal{M}_l$ can be presented as an $\mathbb{F}_q[x]$-linear combination of $P_t(x, y)$, equation (9) forms a basis $\mathcal{B}_l$ of $\mathcal{M}_l$.

B. Module Minimization

In order to describe the minimization process, we need to present $\mathcal{B}_l$ as a matrix over $\mathbb{F}_q[x]$.

Definition II. Let $\underline{\xi} = (\xi_0(x), \xi_1(x), \ldots, \xi_l(x))$ denote a vector over $\mathbb{F}_q[x]$, the degree of $\underline{\xi}$ is $\deg \underline{\xi} = \max\{\deg \xi_\tau(x), \forall \tau\}$. The leading position (LP) of $\underline{\xi}$ is $\mathrm{LP}(\underline{\xi}) = \max\{\tau \mid \deg \xi_\tau(x) = \deg \underline{\xi}\}$. Since $\xi_\tau(x) = \xi_\tau^{(0)} + \xi_\tau^{(1)} x + \cdots + \xi_\tau^{(\deg \xi_\tau(x))} x^{\deg \xi_\tau(x)}$, the leading term (LT) of $\xi_\tau(x)$ is $\mathrm{LT}(\xi_\tau(x)) = \xi_\tau^{(\deg \xi_\tau(x))} x^{\deg \xi_\tau(x)}$.

Given a matrix $\mathcal{V}$ over $\mathbb{F}_q[x]$, let $\mathcal{V}|_t$ denote its row-t and $\mathcal{V}|_t^{(\tau)}$ denote its entry of row-t column-τ. Since $P_t(x,y) = \sum_{\tau \le l} P_t^{(\tau)}(x) y^\tau$ where $P_t^{(\tau)}(x) \in \mathbb{F}_q[x]$, it can be presented as a vector over $\mathbb{F}_q[x]$, i.e., $(P_t^{(0)}(x), P_t^{(1)}(x), \ldots, P_t^{(l)}(x))$. Therefore, $\mathcal{B}_l$ can be presented as an $(l+1) \times (l+1)$ square matrix over $\mathbb{F}_q[x]$, where $\mathcal{B}_l|_t^{(\tau)} = P_t^{(\tau)}(x)$.

Definition III [9]. Given a square matrix $\mathcal{V}$ over $\mathbb{F}_q[x]$, if any of its two rows $\mathcal{V}|_t$ and $\mathcal{V}|_{t'}$ exhibit $\mathrm{LP}(\mathcal{V}|_t) \neq \mathrm{LP}(\mathcal{V}|_{t'})$, it is in the *weak Popov form*.

Now, let us define

$$\mathcal{D}_v = \mathrm{diag}(1, x^{k-1}, \ldots, x^{v(k-1)}) \tag{10}$$

and

$$\mathcal{D}_v^{-1} = \mathrm{diag}(1, x^{-(k-1)}, \ldots, x^{-v(k-1)}). \tag{11}$$

After $\mathcal{B}_l$ is formed by (9), it will be mapped as $\mathcal{A}_l = \mathcal{B}_l \cdot \mathcal{D}_l$, so that $\deg \mathcal{A}_l|_t = \deg_{1,k-1} P_t(x,y)$. Afterwards, the MS algorithm that is shown in Algorithm 1 will reduce $\mathcal{A}_l$ into the weak Popov form $\mathcal{A}'_l$. Demap it as $\mathcal{B}'_l = \mathcal{A}'_l \cdot \mathcal{D}_l^{-1}$, and $\mathcal{B}'_l$ is the Gröbner basis. Polynomials $P'_0(x,y), \ldots, P'_l(x,y)$ can be retrieved from $\mathcal{B}'_l$ by $P_0'^{(\tau)}(x) = \mathcal{B}'_l|_0^{(\tau)}, \ldots, P_l'^{(\tau)}(x) = \mathcal{B}'_l|_l^{(\tau)}$, respectively. Among them, the minimum one is chosen as the interpolated polynomial $Q(x,y)$. Factorize Q to yield the message $f(x)$ [12].

Algorithm 1 The Mulders-Storjohann Algorithm

Input: $\mathcal{A}_l$;

1: **While** $\mathcal{A}_l$ is not in the weak Popov form **do**
2: Find two rows $\mathcal{A}_l|_t$ and $\mathcal{A}_l|_{t'}$ such that $\deg \mathcal{A}_l|_t \le \deg \mathcal{A}_l|_{t'}$ and $\mathrm{LP}(\mathcal{A}_l|_t) = \mathrm{LP}(\mathcal{A}_l|_{t'})$;
3: Perform $\mathcal{A}_l|_{t'} \leftarrow \mathcal{A}_l|_{t'} - \frac{\mathrm{LT}(\mathcal{A}_l|_{t'}^{\mathrm{LP}(\mathcal{A}_l|_{t'})})}{\mathrm{LT}(\mathcal{A}_l|_t^{\mathrm{LP}(\mathcal{A}_l|_t)})} \cdot \mathcal{A}_l|_t$;
4: **End while**

IV. The PASD-MM Algorithm

This section introduces the proposed PASD-MM algorithm. Let v denote the progressive iteration index and $1 \le v \le l$.

A. Submodule and its Image

Submodule is the subspace of a module, which is defined as follows.

Definition IV. Given a module $\mathcal{M}_l$ that is generated by (9), its submodule $\mathfrak{M}_v$ is the subspace spanned by $P_0(x,y) \sim P_v(x,y)$.

Note that since $\deg_y P_v(x,y) \le v$, the basis of $\mathfrak{M}_v$ can be presented as a $(v+1) \times (v+1)$ square matrix over $\mathbb{F}_q[x]$.

For a balanced list L'_j, we define

$$\delta_j(t) = \mathsf{m}_j(t) - \mathsf{m}_j(t+1) \tag{12}$$

and $t = 0, 1, \ldots, l$. Since $\mathsf{m}_j \le l$, $\mathsf{m}_j(l+1) = \mathsf{m}_j(l) = 0$. Consequently, $\delta_j(l-1) = \mathsf{m}_j(l-1)$ and $\delta_j(l) = 0$. Let us further define

$$G_t(x) = \prod_{j=0}^{n-1} (x - \alpha_j)^{\mathsf{m}_j(t)} \tag{13}$$

and

$$R_t(x) = \prod_{j=0}^{n-1} (x - \alpha_j)^{\delta_j(t)}. \tag{14}$$

Based on (12), it can be realized that

$$G_t(x) = G_{t+1}(x) R_t(x). \tag{15}$$

Since $\mathsf{m}_j(l) = \delta_j(l) = 0, \forall j$, $G_l(x) = R_l(x) = 1$.

Let $\{\theta_t^\eta\}$ denote a set of η elements, each of which is distinctively chosen from the integer set $\{0, 1, \ldots, t-1\}$, where $0 \le \eta \le t$. Furthermore, let Θ_t^η denote a collection of all sets $\{\theta_t^\eta\}$ as $\Theta_t^\eta = \{\{\theta_t^\eta\}\}$. Note that $\Theta_t^0 = \emptyset$, $|\Theta_t^\eta| = \binom{t}{\eta}$ and $|\Theta_t^t| = 1$. E.g., if $t = 4$ and $\eta = 2$, $\Theta_4^2 = \{\{0,1\}, \{0,2\}, \{0,3\}, \{1,2\}, \{1,3\}, \{2,3\}\}$. Finally, let $\mathbf{0}_{\iota \times \kappa}$ denote an all zero matrix of size $\iota \times \kappa$.

With the above notations, generators (9) can be rewritten as

$$P_t(x,y) = G_t(x) W_t(x,y), \tag{16}$$

where

$$W_t(x,y) = \prod_{\varepsilon=0}^{t-1} (y - F_\varepsilon(x)) = \sum_{\tau=0}^{t} w_t^{(\tau)}(x) y^\tau \tag{17}$$

and

$$w_t^{(\tau)}(x) = \sum_{\Theta_t^{t-\tau}} \prod_{\varepsilon \in \{\theta_t^{t-\tau}\}} (-F_\varepsilon(x)). \tag{18}$$

Note that $w_t^{(t)}(x) = 1$ and $W_0(x,y) = 1$. The following theorem further characterizes the basis $\mathfrak{B}_v$ of $\mathfrak{M}_v$.

Theorem 2. The matrix representation of $\mathfrak{B}_v$ can be written as

$$\mathfrak{B}_v = G_v(x) \cdot \Xi_v, \tag{19}$$

where

$$\Xi_v = \begin{bmatrix} R_{v-1}(x) \cdot \Xi_{v-1} & \mathbf{0}_{v \times 1} \\ w_v^{(0)}(x) \quad \cdots \quad w_v^{(v-1)}(x) & w_v^{(v)}(x) \end{bmatrix} \tag{20}$$

is the image of $\mathfrak{B}_v$.

Proof: Based on (16) – (18), when $v = 1$, $\mathfrak{B}_1$ contains

$$P_0(x,y) = G_0(x),$$

$$P_1(x,y) = G_1(x) W_1(x,y).$$

Since $G_0(x) = G_1(x) R_0(x)$,

$$\mathfrak{B}_1 = G_1(x) \cdot \Xi_1,$$

where

$$\Xi_1 = \begin{bmatrix} R_0(x) & 0 \\ w_1^{(0)}(x) & w_1^{(1)}(x) \end{bmatrix}.$$

Similarly, $\mathfrak{B}_2$ can be written as

$$\mathfrak{B}_2 = G_2(x) \cdot \Xi_2,$$

where

$$\Xi_2 = \begin{bmatrix} R_1(x)R_0(x) & 0 & 0 \\ R_1(x)w_1^{(0)}(x) & R_1(x)w_1^{(1)}(x) & 0 \\ w_2^{(0)}(x) & w_2^{(1)}(x) & w_2^{(2)}(x) \end{bmatrix} = \begin{bmatrix} R_1(x) \cdot \Xi_1 & \mathbf{0}_{2\times 1} \\ w_2^{(0)}(x) & w_2^{(1)}(x) \quad w_2^{(2)}(x) \end{bmatrix}.$$

Continuing the deduction, $\mathfrak{B}_v$ can be written as in (19). ■

Based on Theorem 2 and since $G_l(x) = 1$, we have $\mathfrak{B}_l = \Xi_l$. Further recalling Definition IV, we have $\mathfrak{B}_l = \mathcal{B}_l$. Therefore, the following corollary can be led to.

Corollary 3. When $v = l$, $\mathfrak{B}_l = \Xi_l = \mathcal{B}_l$.

Theorem 2 and Corollary 3 reveal that the basis $\mathcal{B}_l$ can be progressively constructed through its images of submodule's basis. Recalling Algorithm 1, the MS algorithm performs multiple $\mathbb{F}_q[x]$-linear combinations between rows of $\mathcal{A}_l$. This row operations can be rescheduled as follows. The MS algorithm can target the first two rows of $\mathcal{A}_l$. This is equivalent to reducing matrix $\mathfrak{B}_1 \cdot \mathcal{D}_1$ into the weak Popov form. Afterwards, it can further target the first three rows of $\mathcal{A}_l$, which is equivalent to reducing matrix $\mathfrak{B}_2 \cdot \mathcal{D}_2$ into the weak Popov form. Continue the process until matrix $\mathfrak{B}_l \cdot \mathcal{D}_l$ is in the weak Popov form, and $\mathcal{A}'_l = \mathfrak{B}_l \cdot \mathcal{D}_l$. Since $G_v(x)$ is the GCD of all bivariate polynomials of $\mathfrak{B}_v$, performing the MS algorithm on $\mathfrak{B}_v \cdot \mathcal{D}_v$ is equivalent to performing it on $\Xi_v \cdot \mathcal{D}_v$. The following PASD-MM algorithm is proposed based on the above observation. It aims to decode the message from an intermediate interpolated polynomial $Q_v(x, y)$ where $\deg_y Q_v = v$. The polynomial is retrieved from the weak Popov form matrix $\Xi_v \cdot \mathcal{D}_v$.

B. The PASD-MM Algorithm

The algorithm aims to decode the message from one of the progressively enlarged images of submodule's basis. At the beginning, $v = 1$, image Ξ_1 is initialized as

$$\mathcal{P}_{1,0}(x, y) = R_0(x), \tag{21}$$

$$\mathcal{P}_{1,1}(x, y) = W_1(x, y). \tag{22}$$

Map Ξ_1 into $\mathcal{X}_1 = \Xi_1 \cdot \mathcal{D}_1$ and the MS algorithm will reduce $\mathcal{X}_1$ into the weak Popov form $\mathcal{X}'_1$. Demap it as $\Xi'_1 = \mathcal{X}'_1 \cdot \mathcal{D}_1^{-1}$. Polynomials $\mathcal{P}'_{1,0}(x, y)$ and $\mathcal{P}'_{1,1}(x, y)$ can be retrieved from Ξ'_1 by $\mathcal{P}'^{(\tau)}_{1,0}(x) = \Xi'_1|_0^{(\tau)}$ and $\mathcal{P}'^{(\tau)}_{1,1}(x) = \Xi'_1|_1^{(\tau)}$, respectively. Among them, the minimum one is chosen as the interpolated polynomial $Q_1(x, y)$. Further factorize Q_1 to determine its y-root. If $Q_1(x, f(x)) = 0$, the message is found and the decoding terminates. Otherwise, the decoding progresses to determine $Q_2(x, y)$ where $\deg_y Q_2 = 2$. Image Ξ_2 needs to be constructed. Based on Theorem 2, it can be generated by

$$\mathcal{P}_{2,0}(x, y) = R_1(x)\mathcal{P}'_{1,0}(x, y), \tag{23}$$

$$\mathcal{P}_{2,1}(x, y) = R_1(x)\mathcal{P}'_{1,1}(x, y), \tag{24}$$

$$\mathcal{P}_{2,2}(x, y) = W_2(x, y). \tag{25}$$

Again, generate $\mathcal{X}_2 = \Xi_2 \cdot \mathcal{D}_2$ and the MS algorithm will reduce $\mathcal{X}_2$ into the weak Popov form $\mathcal{X}'_2$. Demap it as $\Xi'_2 = \mathcal{X}'_2 \cdot \mathcal{D}_2^{-1}$. Polynomials $\mathcal{P}'_{2,0}(x, y)$, $\mathcal{P}'_{2,1}(x, y)$ and $\mathcal{P}'_{2,2}(x, y)$ can be retrieved from Ξ'_2 by $\mathcal{P}'^{(\tau)}_{2,0}(x) = \Xi'_2|_0^{(\tau)}$, $\mathcal{P}'^{(\tau)}_{2,1}(x) = \Xi'_2|_1^{(\tau)}$ and $\mathcal{P}'^{(\tau)}_{2,2}(x) = \Xi'_2|_2^{(\tau)}$, respectively. Among them, the minimum one is chosen as $Q_2(x, y)$. Again, if $Q_2(x, f(x)) = 0$, the message is found and the decoding terminates. Otherwise, the decoding continues.

In general, at progressive iteration $v - 1$ ($v \geq 2$), if Ξ'_{v-1} is the image produced by the MS algorithm, polynomials $\mathcal{P}'_{v-1,0}(x, y), \ldots, \mathcal{P}'_{v-1,v-1}(x, y)$ are retrieved from Ξ'_{v-1} by $\mathcal{P}'^{(\tau)}_{v-1,0}(x) = \Xi'_{v-1}|_0^{(\tau)}, \ldots, \mathcal{P}'^{(\tau)}_{v-1,v-1}(x) = \Xi'_{v-1}|_{v-1}^{(\tau)}$, respectively. Among them, the minimum one is chosen as $Q_{v-1}(x, y)$. If $Q_{v-1}(x, f(x)) = 0$, the message is found and the decoding terminates. Otherwise, Ξ'_{v-1} will be expanded to Ξ_v in order to find $Q_v(x, y)$ where $\deg_y Q_v = v$. Based on Theorem 2, Ξ_v can be generated by

$$\mathcal{P}_{v,t}(x, y) = R_{v-1}(x)\mathcal{P}'_{v-1,t}(x, y), \text{ if } 0 \leq t \leq v - 1, \tag{26}$$

$$\mathcal{P}_{v,v}(x, y) = W_v(x, y). \tag{27}$$

Based on (17) and (18), we know $\mathcal{P}_{v,v}(x, y)$ can be directly generated based on the balanced lists. It does not need to know the intermediate decoding information. After generating Ξ_v, it will be mapped by

$$\mathcal{X}_v = \Xi_v \cdot \mathcal{D}_v. \tag{28}$$

The MS algorithm will then reduce $\mathcal{X}_v$ into the weak Popov form $\mathcal{X}'_v$. Further demap it as

$$\Xi'_v = \mathcal{X}'_v \cdot \mathcal{D}_v^{-1}. \tag{29}$$

Polynomials $\mathcal{P}'_{v,0}(x, y), \ldots, \mathcal{P}'_{v,v}(x, y)$ can be retrieved from Ξ'_v by $\mathcal{P}'^{(\tau)}_{v,0}(x) = \Xi'_v|_0^{(\tau)}, \ldots, \mathcal{P}'^{(\tau)}_{v,v}(x) = \Xi'_v|_v^{(\tau)}$, respectively. Among them, the minimum one is chosen as the interpolated polynomial $Q_v(x, y)$. Further factorize Q_v. If $Q_v(x, f(x)) = 0$, the decoding terminates. Otherwise, the decoding progresses by updating $v = v + 1$. If $v > l$, it implies the designed maximum y-degree of the interpolated polynomial is exceeded. The decoding also terminates with a decoding failure. Otherwise, the above decoding continues.

The PASD-MM proposal is summarized as follows.

Algorithm 2 The PASD-MM Algorithm

Input: M;

1: Generate all balanced lists L'_j as in (6);
2: Initialize $v = 1$ and $\mathcal{P}'_{0,0}(x, y) = 1$;
3: Formulate Ξ_v as in (26) and (27);
4: Generate $\mathcal{X}_v$ by (28);
5: Perform Algorithm 1, yielding $\mathcal{X}'_v$;
6: Generate Ξ'_v by (29) to find polynomial $Q_v(x, y)$;
7: Factorize Q_v. If $Q_v(x, f(x)) = 0$, output $f(x)$ and terminate the decoding; Otherwise, update $v = v + 1$;
8: If $v > l$, terminate the decoding; Otherwise, go to **3**.

TABLE I
AVERAGE COMPLEXITY IN DECODING THE (255, 239) RS CODE

SNR (dB)	5.0	5.5	6.0	6.5	7.0	7.5	8.0
ASD	1.08×10^9	1.11×10^9	1.11×10^9	1.08×10^9	1.01×10^9	9.48×10^8	8.89×10^8
PASD	2.39×10^8	1.38×10^8	3.48×10^7	1.13×10^7	1.10×10^7	1.09×10^7	1.09×10^7
ASD-MM	2.92×10^7	3.02×10^7	2.82×10^7	2.69×10^7	2.57×10^7	2.43×10^7	2.31×10^7
PASD-MM	2.05×10^7	1.30×10^7	3.23×10^6	8.29×10^5	6.94×10^5	6.89×10^5	6.87×10^5

V. PERFORMANCE AND COMPLEXITY ANALYSES

This section provides the decoding frame error rate (FER) and complexity performances of the PASD-MM algorithm. The complexity is measured as the average number of finite field arithmetic operations in decoding a codeword. Our results are obtained over the additive white Gaussian noise (AWGN) channel using BPSK modulation.

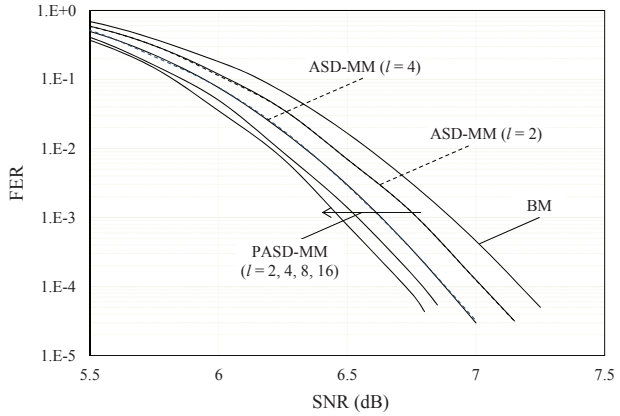

Fig. 1. Performance of the (255, 239) RS code over the AWGN channel.

Fig. 1 shows the PASD-MM performance for the popular (255, 239) RS code. It outperforms the conventional BM algorithm, where the performance gain can be improved by increasing the decoding parameter l, i.e., $\deg_y Q$. This will be at the cost of decoding complexity. Meanwhile, our simulation results also verify that with the same decoding parameter, the PASD-MM algorithm can achieve the same performance as the ASD-MM algorithm. It demonstrates that the progressive decoding maintains the error-correction capability of its non-progressive prototype.

Table I further shows the average complexity in decoding the RS code. For all algorithms, $l = 4$. Note that the ASD and PASD algorithms employ Koetter's interpolation. It shows both of the progressive decoding algorithms are less complex than their non-progressive prototypes. Although multiple factorizations may be performed in the progressive decoding, their complexity are marginal and can be compensated by the progressive interpolation feature. As the signal-to-noise ratio (SNR) increases, the received information becomes more reliable so that the message can be decoded with a smaller l, resulting in a smaller complexity. On the other hand, the ASD-MM and PASD-MM algorithms are less complex than their counterparts by at least an order of magnitude. This thanks to the MM interpolation technique. Among the four algorithms, the PASD-MM algorithm is the simplest. When SNR = 7 dB, most of its decoding events terminate with $l = 1$, resulting in a complexity that is far smaller than the other algorithms. For this RS code, the BM complexity is 4.44×10^4. Again, it should be emphasized that the PASD-MM algorithm removes the memory requirement of the original PASD algorithm, becoming more friendly for implementation.

VI. CONCLUSION

This paper has introduced the PASD-MM algorithm for RS codes. It exhibits an advanced decoding performance and a low complexity that is also channel dependent. It generates the interpolated polynomial with a progressively enlarged y-degree. Using the MM interpolation, this can be realized through the progressively enlarged images of submodule's basis. This progressive decoding is realized without any memory requirement, making it more friendly for implementation.

ACKNOWLEDGEMENT

This work is sponsored by the National Natural Science Foundation of China (NSFC) with project ID 61671486.

REFERENCES

[1] J. Massey, "Shift register synthesis and BCH decoding," *IEEE Trans. Inform. Theory*, vol. 15, no. 1, pp. 122–127, Jan. 1969.
[2] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraic-geometric codes," *IEEE Trans. Inform. Theory*, vol. 45, no. 6, pp. 1757–1767, Sept. 1999.
[3] R. Koetter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," *IEEE Trans. Inform. Theory*, vol. 49, no. 11, pp. 2809–2825, Nov. 2003.
[4] R. Koetter, "On algebraic decoding of algebraic-geometric and cyclic codes," Ph.D. dissertation, Univ. Linköping, Linköping, Sweden, 1996.
[5] R. Koetter, 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] Y. Wu, "New list decoding algorithms for Reed-Solomon and BCH codes," *IEEE Trans. Inform. Theory*, vol. 54, no. 8, pp. 3611–3630, Aug. 2008.
[7] 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.
[8] 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.
[9] T. Mulders and A. Storjohann, "On lattice reduction for polynomial matrices," *J. Symb. Comput.*, vol. 35, no. 4, pp. 377–401, Apr. 2003.
[10] M. Alekhnovich, "Linear Diophantine equations over polynomials and soft decoding of Reed-Solomon codes," *IEEE Trans. Inform. Theory*, vol. 51, no. 7, pp. 2257–2265, Jul. 2005.
[11] T. Kaneko, T. Nishijima, H. Inazumi, and S. Hirasawa, "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.
[12] R. Roth and G. Ruckenstein, "Efficient decoding of Reed-Solomon codes beyond half the minimum distance," *IEEE Trans. Inform. Theory*, vol. 46, no. 1, pp. 246–257, Jan. 2000.