Progressive Algebraic Soft-Decision Decoding of Reed-Solomon Codes

Li Chen, *Member, IEEE*, Siyun Tang, and Xiao Ma, *Member, IEEE*

***Abstract*—The algebraic soft-decision decoding (ASD) algorithm is a polynomial-time soft decoding algorithm for Reed-Solomon (RS) codes. It outperforms both the algebraic hard-decision decoding (AHD) and the conventional unique decoding algorithms, but with a high computational cost. This paper proposes a progressive ASD (PASD) algorithm that enables the conventional ASD algorithm to perform decoding with an adjustable designed factorization output list size (OLS). The OLS is enlarged progressively leading to an incremental computation for the interpolation and an enhanced error-correction capability. Multiple factorizations are performed in order to find out the intended message polynomial which will be validated by a cyclic redundant check (CRC) code. The incremental interpolation constraints are introduced to characterize the progressive decoding. The validity analysis of the algorithm shows the PASD algorithm is a natural and computationally saving generalization of the ASD algorithm, delivering the same interpolation solution. The average decoding complexity of the algorithm is further theoretically characterized, revealing its dependence on the channel condition. The simulation results further validate the analysis by showing that the average decoding complexity can be converged to the minimal level in a good channel condition. Finally, performance evaluation shows the PASD algorithm preserves the error-correction capability of the ASD algorithm.**

***Index Terms*—Algebraic soft-decision decoding, complexity reduction, Koetter-Vardy algorithm, progressive interpolation, Reed-Solomon codes.**

I. Introduction

REED-Solomon (RS) codes are widely used in the communication systems and storage devices for error-correction. The conventional unique decoding algorithms [1] - [3] are efficient, but their error-correction capability is limited by the half Hamming distance bound. The generalized minimum-distance (GMD) decoding [4] [5] improves the error-correction performance. The recent algebraic hard-decision decoding (AHD) (or the so called list decoding) algorithm [6] [7] can correct errors beyond the distance bound. By introducing a reliability transform front-end to the AHD algorithm, the algebraic soft-decision decoding (ASD) algorithm [8] further achieves a significant performance gain over the AHD and GMD algorithms. Unlike other soft-decoding algorithms [9] [10] for RS codes, the ASD algorithm is a polynomial-time soft decoding algorithm that inherits a moderate decoding complexity. However, it is still orders of magnitude higher than that of the unique decoding algorithms, bringing an expensive implementation cost.

The ASD algorithm's high decoding complexity is mainly caused by the interpolation process [11] [12]. So far, various interpolation complexity reduction approaches exist. In [13] - [16], coordinate transform of the interpolation points were introduced. They are either transformed into a set of points with a zero y-coordinate [13] - [15], or formed with the knowledge of the error locator and the error-correction polynomials that are yielded by the Berlekamp-Massey (BM) algorithm [16]. In [17], a Chase type interpolation was introduced. Interpolation complexity can be reduced by incorporating the coordinate transform of [14] and identifying the common interpolation points of various test received vectors. In [11] [18], the interpolated polynomials with a leading order greater than the interpolation cost were eliminated during the interpolation, reducing the unnecessary decoding computation and system memory. While in [19], interpolation is sequentially performed on the polynomial that is most likely to be chosen for factorization. It avoids spending computations on those unlikely interpolated polynomials. More than reducing the interpolation complexity, other main complexity reduction approaches include a facilitated reliability transform [20] and a hybrid decoding system [21].

However, the complexity reduction potential of the ASD algorithm has not been fully exploited. The ASD algorithm is flexible in nature since decoding with a different designed factorization output list size (OLS) enables the algorithm to have a different error-correction capability and complexity. It implies the ASD algorithm can be performed through iterations during which the designed OLS is enlarged progressively. That will result in a gradually enhanced error-correction capability and the incremental computations between iterations. Consequently, both of the algorithm's error-correction capability and decoding complexity are channel dependent. Motivated by the fact that different decoding events require different error-correction capabilities, this paper proposes the progressive ASD (PASD) algorithm in which the original ASD algorithm is performed through iterations enlarging the designed OLS gradually. The original reliability transform algorithm [8] is modified to generate a series of multiplicity matrices w.r.t. different designed OLS values. The incremental interpolation constraints are further introduced in order to characterize the progressive interpolation algorithm. Factorization is performed after each round of interpolation in an attempt to find out the

Paper approved by E. Ayanoglu, the Editor for Coding and Communication Theory of the IEEE Communications Society. Manuscript received November 7, 2011; revised May 31, 2012.

The authors are with the School of Information Science and Technology, Sun Yat-sen University, Guangzhou, China, 510006 (e-mail: chenli55@mail.sysu.edu.cn; website: sist.sysu.edu.cn/∼chenli.

This work is sponsored by the National Natural Science Foundation of China (NSFC) with project ID 61001094, the National Basic Research Program of China (973 program) with project ID 2012CB316100, and the Guangdong Natural Science Foundation (GDNSF) with project ID 10451027501005078.

Digital Object Identifier 10.1109/TCOMM.2012.100912.110752

message polynomial which will be validated by a cyclic redundancy check (CRC) code. It enables the PASD algorithm to adjust its error-correction capability and decoding complexity to the quality of the received information. The validity analysis of the algorithm proves it is a natural and computationally saving generalization of the original interpolation problem. Decoding with the same designed OLS, the PASD algorithm only reschedules the interpolation order without affecting the final solution. The average decoding complexity of the algorithm is further theoretically characterized, unveiling its dependence on the channel condition. Simulation results show that more significant complexity reduction can be achieved by improving the channel condition. Finally, the performance evaluation shows both the PASD and ASD algorithms have a similar error-correction performance.

The rest of the paper is organized as the follows: Section II shows some preliminaries for the paper. Section III presents details of the PASD algorithm. Section IV analyzes the validity of the algorithm. Section V and VI present the complexity analysis and performance evaluation for the PASD algorithm, respectively. Finally, Section VII concludes the paper.

II. Preliminaries

Let $\mathbb{F}_q = \{\alpha_0, \alpha_1, \ldots, \alpha_{q-1}\}$ denote the finite field of size q, $\mathbb{F}_q[x]$ and $\mathbb{F}_q[x, y]$ denote the rings of univariate and bivariate polynomials defined over $\mathbb{F}_q$, respectively. Given a message vector $(u_0, u_1, \ldots, u_{k-1}) \in \mathbb{F}_q^k$, the message polynomial $u(x) \in \mathbb{F}_q[x]$ can be written as:

$$u(x) = u_0 + u_1 x + \cdots + u_{k-1} x^{k-1}. \tag{1}$$

In the PASD algorithm, the message in bits will need to be CRC encoded first. Then the coded bits are converted to the message vector as shown above. The codeword $\bar{c}$ of an (n, k) RS code can be generated by:

$$\bar{c} = (c_0, c_1, \ldots, c_{n-1}) = (u(x_0), u(x_1), \ldots, u(x_{n-1})), \tag{2}$$

where $\bar{c} \in \mathbb{F}_q^n$ and $x_0, x_1, \ldots, x_{n-1}$ are n distinct nonzero elements of $\mathbb{F}_q$.

Assuming $\bar{c}$ is transmitted through a discrete memoryless channel and $\bar{y} = (y_0, y_1, \ldots, y_{n-1}) \in \mathbb{R}^n$ is the received vector. The ASD algorithm will first obtain a reliability matrix Π of size $q \times n$ based on $\bar{y}$. Entry π_{ij} of Π is the *a posteriori* probability value which is defined as:

$$\pi_{ij} = \Pr[c_j = \alpha_i \mid y_j]. \tag{3}$$

The reliability matrix Π is then transformed into a multiplicity matrix M of the same size [8]. Entry m_{ij} represents the interpolation multiplicity for the point (x_j, α_i). The interpolation is to construct a bivariate polynomial $Q \in \mathbb{F}_q[x, y]$ that passes through all the points with different multiplicities which are indicated by the matrix M. Given a polynomial $Q = \sum_{a,b} Q_{ab} x^a y^b$, its Hasse derivative evaluation at the point (x_j, α_i) can be defined as [22]:

$$\mathcal{D}_{r,s}(Q(x,y))|_{x=x_j, y=\alpha_i} = \sum_{a \geq r, b \geq s} \binom{a}{r} \binom{b}{s} Q_{ab} x_j^{a-r} \alpha_i^{b-s}, \tag{4}$$

where (r, s) is a pair of nonnegative integers. To have a zero of multiplicity m_{ij} at (x_j, α_i), Q must satisfy $\mathcal{D}_{r,s}(Q(x,y))|_{x=x_j, y=\alpha_i} = 0$ for all $r + s < m_{ij}$. To clarify the interpolation constraints, the following definition is given.

Definition 1: Let $\Lambda(m_{ij})$ denote the set of interpolation constraints defined by m_{ij} as:

$$\Lambda(m_{ij}) = \{\mathcal{D}_{r,s}(Q(x,y))|_{x=x_j, y=\alpha_i} = 0, \ \forall \ r + s < m_{ij}\}. \tag{5}$$

Then $\Lambda(\mathrm{M})$ denotes a collection of all the sets $\Lambda(m_{ij})$ defined by the entries of M as:

$$\Lambda(\mathrm{M}) = \{\Lambda(m_{ij}), \ \forall \ m_{ij} \in \mathrm{M}\}. \tag{6}$$

Since the number of interpolation constraints defined by m_{ij} is $|\Lambda(m_{ij})| = \binom{m_{ij}+1}{2} = (m_{ij} + 1)m_{ij}/2$, the total number of constraints defined by matrix M is given by:

$$\mathcal{C}(\mathrm{M}) = |\Lambda(\mathrm{M})| = \frac{1}{2} \sum_{i=0}^{q-1} \sum_{j=0}^{n-1} m_{ij}(m_{ij} + 1), \tag{7}$$

which is also called the interpolation cost. Since the interpolation constraints can be uniquely defined by (r, s) and the point (x_j, α_i), in the followings of the paper, we simply use $(r, s)_{ij}$ to denote the interpolation constraints $\mathcal{D}_{r,s}(Q(x,y))|_{x=x_j, y=\alpha_i} = 0$ and $\Lambda(m_{ij}) = \{(r, s)_{ij}, \ \forall \ r + s < m_{ij}\}$. Moreover, the Hasse derivative evaluation (4) is denoted by $\mathcal{D}_{(r,s)_{ij}}(Q(x,y))$.

Monomials $x^a y^b$ are organized by the $(1, k-1)$-revlex order [1]. Given a polynomial $Q \in \mathbb{F}_q[x, y]$, if $x^{a'} y^{b'}$ is the leading monomial with $Q_{a'b'} \neq 0$, the $(1, k-1)$-weighted degree of Q is defined as $\deg_{1,k-1} Q = \deg_{1,k-1} x^{a'} y^{b'}$ and its leading order is defined as $\mathrm{lod}(Q) = \mathrm{ord}(x^{a'} y^{b'})$. Given two polynomials $(P, Q) \in \mathbb{F}_q[x, y]$, we can declare $P \leq Q$ if $\mathrm{lod}(P) \leq \mathrm{lod}(Q)$.

For matrices Π and M, we use i_j to denote the row index at column j that yields $\alpha_{i_j} = c_j$ as:

$$i_j = \mathrm{index}\{\alpha_i \mid \alpha_i = c_j\}. \tag{8}$$

We further define the multiplicity based codeword score as:

$$S_{\mathrm{M}}(\bar{c}) = \sum_{j=0}^{n-1} m_{i_j j}, \tag{9}$$

and the reliability based codeword score as:

$$S_{\Pi}(\bar{c}) = \sum_{j=0}^{n-1} \pi_{i_j j}. \tag{10}$$

With (8), we know the codeword $\bar{c} = (\alpha_{i_0}, \alpha_{i_1}, \ldots, \alpha_{i_{n-1}})$. Hence, $S_{\Pi}(\bar{c})$ represents the sum of the reliability values of all codeword symbols. Therefore, given a codeword $\bar{c}$ and two distinct reliability matrices Π_1 and Π_2, we claim Π_1 has a better quality if $S_{\Pi_1}(\bar{c}) > S_{\Pi_2}(\bar{c})$.

According to Lemma 4 of [8], if the message polynomial $u(x)$ evaluates to the codeword as in (2), then $(x - x_0)^{m_{i_0 0}} (x - x_1)^{m_{i_1 1}} \cdots (x - x_{n-1})^{m_{i_{n-1} n-1}} \mid Q(x, u(x))$. Since $S_{\mathrm{M}}(\bar{c})$ is

[1]The $(1, k-1)$-weighted degree of monomial $x^a y^b$ is defined as $\deg_{1,k-1} x^a y^b = a + (k-1)b$. Given two distinct monomials $x^{a_1} y^{b_1}$ and $x^{a_2} y^{b_2}$, we have $\mathrm{ord}(x^{a_1} y^{b_1}) < \mathrm{ord}(x^{a_2} y^{b_2})$, if $\deg_{1,k-1} x^{a_1} y^{b_1} < \deg_{1,k-1} x^{a_2} y^{b_2}$, or $\deg_{1,k-1} x^{a_1} y^{b_1} = \deg_{1,k-1} x^{a_2} y^{b_2}$ and $b_1 < b_2$.

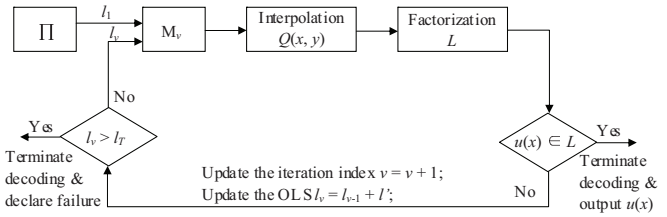

Fig. 1. Block diagram of the PASD decoder.

the degree of $(x-x_0)^{m_{i_0 0}}(x-x_1)^{m_{i_1 1}}\cdots(x-x_{n-1})^{m_{i_{n-1}n-1}}$ and $\deg Q(x,u(x)) = \deg_{1,k-1} Q(x,y)$, if

$$S_{\mathrm{M}}(\bar{c}) > \deg_{1,k-1} Q(x,y), \tag{11}$$

then $Q(x,u(x)) = 0$ and the message polynomial $u(x)$ can be found out by factorizing $Q(x,y)$ [7] [8] [11]. Factorization [7] [25] [26] [27] is to find out a list of output candidates $p(x)$ that are in the form of $u(x)$,

$$L = \{p(x) : Q(x,p(x)) = 0\}. \tag{12}$$

The cardinality of L is referred as the factorization OLS. Since $\deg_{1,k-1} Q(x,y)$ is upper bounded by $\Delta_{1,k-1}(\mathcal{C}(\mathrm{M}))$ and $\Delta_{1,k-1}(\mathcal{C}(\mathrm{M})) = \deg_{1,k-1}(x^a y^b | \mathrm{ord}(x^a y^b) = \mathcal{C}(\mathrm{M}))$ [23], the message polynomial $u(x)$ can also be found out if

$$S_{\mathrm{M}}(\bar{c}) > \Delta_{1,k-1}(\mathcal{C}(\mathrm{M})). \tag{13}$$

III. Progressive Algebraic Soft-Decision Decoding

This section will introduce the PASD algorithm, including the decoder block diagram, the progressive reliability transform algorithm and the progressive interpolation algorithm.

A. Decoder Block Diagram

Fig.1 shows the block diagram of the PASD decoder, and it is an iterative process. Let v denote the iteration index which is initialized as $v = 1$ at the beginning. Let l_v denote the designed factorization OLS of iteration v, which is updated during the iterations as:

$$l_v = l_{v-1} + l', \tag{14}$$

where l' is the step size for the designed OLS update. Unless otherwise stated, it is assumed that $l_1 = 1$ and $l' = 1$ in this paper. A designed maximal OLS is denoted by l_T, indicating the maximal decoding complexity that the decoder tolerates. Unlike the ASD algorithm, the PASD algorithm performs decoding with the following series of designed OLS values:

$$l_1, l_2, \ldots, l_{v-1}, l_v, \ldots, l_T. \tag{15}$$

Based on the series of designed OLS values, a series of multiplicity matrices are generated by the reliability transform algorithm as:

$$\mathrm{M}_1, \mathrm{M}_2, \ldots, \mathrm{M}_{v-1}, \mathrm{M}_v, \ldots, \mathrm{M}_T. \tag{16}$$

Matrix M_v denotes the multiplicity matrix that corresponds to l_v. During the generation of the multiplicity matrices, entries m_{ij} are increased monotonically. Hence

$$\mathcal{C}(\mathrm{M}_1) < \mathcal{C}(\mathrm{M}_2) < \cdots < \mathcal{C}(\mathrm{M}_{v-1}) < \mathcal{C}(\mathrm{M}_v) < \cdots < \mathcal{C}(\mathrm{M}_T). \tag{17}$$

At the beginning of the PASD algorithm, interpolation will be performed based on matrix M_1 with $\mathcal{C}(\mathrm{M}_1)$ constraints. Factorization will then be performed to determine if $u(x) \in L$, which is validated by the CRC code. If $u(x)$ has not been found, the decoder will update its designed OLS as (14) and deploy the progressive reliability transform algorithm to generate M_2. Interpolation will now be performed with the incremental interpolation constraints such that all constraints defined by M_2 are satisfied. The decoding continues until the message polynomial is found, or the designed maximal OLS l_T is exceeded. The latter case represents a decoding failure.

B. Progressive Reliability Transform

The reliability transform algorithm (algorithm A of [8]) is an iterative algorithm that transfers the reliability values into the multiplicity values. In this paper, it is modified to generate a series of multiplicity matrices of (16) progressively. Since they all correspond to a designed OLS, the algorithm will stop once a designed OLS of l_v is reached, yielding an intermediate multiplicity matrix M_v. Since the outputs are the y-roots of polynomial $Q(x,y)$, the designed OLS is:

$$l_v = \left\lfloor \frac{\Delta_{1,k-1}(\mathcal{C}(\mathrm{M}_v))}{k-1} \right\rfloor. \tag{18}$$

It provides an intermediate stopping criterion for the reliability transform algorithm, delivering a multiplicity matrix M_v that corresponds to a designed OLS of l_v. Note that the actual factorization OLS in each decoding event might be smaller than this value. Unless otherwise stated, we will simply use the OLS to stand for the designed OLS in the rest of the paper.

During this progressive transform, the original reliability matrix Π is preserved. Let Π_v^* denote the intermediate reliability matrix of iteration v. Matrices Π_v^* and M_v are initialized as $\Pi_0^* = \Pi$ and $\mathrm{M}_0 = [0]_{q\times n}$ that is an all-zero matrix. At iteration v, matrices Π, Π_{v-1}^*, M_{v-1} and l_v are taken as the inputs to the reliability transform algorithm. Let $\Pi_v^* = \Pi_{v-1}^*$ and $\mathrm{M}_v = \mathrm{M}_{v-1}$. The algorithm will then be performed until l_v is reached, delivering Π_v^* and M_v.

C. Progressive Interpolation

The interpolation process is implemented by Koetter's algorithm [12]. In order to explicitly describe the progressive interpolation, the incremental interpolation constraints will be first defined as the follows.

Definition II: Let $m_{ij}^{(v-1)}$ and $m_{ij}^{(v)}$ denote the entries of the matrices M_{v-1} and M_v, respectively, the incremental interpolation constraints introduced by the two matrices are defined as a collection of all the residual sets between $\Lambda(m_{ij}^{(v)})$ and $\Lambda(m_{ij}^{(v-1)})$:

$$\Lambda(\Delta\mathrm{M}_v) = \{\Lambda(\mathrm{M}_v) \setminus \Lambda(\mathrm{M}_{v-1})\} = \{\Lambda(m_{ij}^{(v)}) \setminus \Lambda(m_{ij}^{(v-1)})\}. \tag{19}$$

Note that $|\Lambda(\Delta \mathrm{M}_v)| = \mathcal{C}(\mathrm{M}_v) - \mathcal{C}(\mathrm{M}_{v-1})$ and the interpolation at iteration v is to be performed w.r.t. the incremental constraints defined by $\Lambda(\Delta \mathrm{M}_v)$. Also be aware that since $\mathrm{M}_0 = [0]_{q\times n}$ and $\Lambda(\mathrm{M}_0) = \emptyset$, $\Lambda(\Delta \mathrm{M}_1) = \Lambda(\mathrm{M}_1)$. Based on the above introduction, it can be realized that:

$$\Lambda(\mathrm{M}_T) = \Lambda(\Delta \mathrm{M}_1) \cup \Lambda(\Delta \mathrm{M}_2) \cup \cdots \cup \Lambda(\Delta \mathrm{M}_v) \cup \cdots \cup \Lambda(\Delta \mathrm{M}_T). \tag{20}$$

The proposed algorithm is to perform interpolation w.r.t. the constraints of $\Lambda(\Delta \mathrm{M}_1)$, $\Lambda(\Delta \mathrm{M}_2), \ldots, \Lambda(\Delta \mathrm{M}_v), \ldots, \Lambda(\Delta \mathrm{M}_T)$ progressively and determine whether the message polynomial can be found from any of the intermediate interpolation results.

At the beginning of the progressive interpolation, a group of polynomials is initialized:

$$\mathbf{G}_1 = \{g_0, g_1, \ldots, g_{l_1}\} = \{1, y, \ldots, y^{l_1}\}. \tag{21}$$

For each constraint $(r,s)_{ij} \in \Lambda(\Delta \mathrm{M}_1)$, the Hasse derivative evaluation will be performed for each polynomial of $\mathbf{G}_1$. Among those polynomials that do not satisfy the current constraint, the minimal one will be selected as:

$$f_{(r,s)_{ij}} = \min\{g_t \mid \mathcal{D}_{(r,s)_{ij}}(g_t) \neq 0\}. \tag{22}$$

Then, the polynomials of $\mathbf{G}_1$ will be updated according to the following rules [12]:

$$g_t = \begin{cases} g_t, & \text{if } \mathcal{D}_{(r,s)_{ij}}(g_t) = 0 \\ [g_t, f_{(r,s)_{ij}}]_D, & \text{if } \mathcal{D}_{(r,s)_{ij}}(g_t) \neq 0 \text{ and } g_t \neq f_{(r,s)_{ij}} \\ [xg_t, g_t]_D, & \text{if } \mathcal{D}_{(r,s)_{ij}}(g_t) \neq 0 \text{ and } g_t = f_{(r,s)_{ij}}, \end{cases} \tag{23}$$

where $[g,f]_D = \mathcal{D}_{(r,s)_{ij}}(f)g - \mathcal{D}_{(r,s)_{ij}}(g)f$ denotes the bilinear modification for the polynomials of $\mathbb{F}_q[x,y]$. The selected minimal polynomials $f_{(r,s)_{ij}}$ will have to be stored in memory along with their corresponding constraints $(r,s)_{ij}$. They will be reused in the following iterations.

Without loss of generality, we describe the following interpolation process as being performed at iteration v based on the intermediate results of the previous iteration. The progressive interpolation process can be viewed as a *polynomial group expansion* process that yields $\mathbf{G}_v$ based on $\mathbf{G}_{v-1}$. As a result, each polynomial of $\mathbf{G}_v$ satisfies the constraints of $\Lambda(\mathrm{M}_v)$ and has a maximal y-degree of l_v. Let $\widetilde{\mathbf{G}}_{v-1} = \{\tilde{g}_0, \tilde{g}_1, \ldots, \tilde{g}_{l_{v-1}}\}$ denote the updated polynomial group of iteration $v-1$. Each of its polynomials satisfies the constraints of $\Lambda(\mathrm{M}_{v-1})$. The *polynomial group expansion* at iteration v consists of two successive steps that are described below.

Expansion I: Expand the number of polynomials of the group from $l_{v-1}+1$ to l_v+1 and enable each polynomial of the expanded group to satisfy the constraints defined by $\Lambda(\mathrm{M}_{v-1})$. The incremental polynomial group $\Delta\mathbf{G}_v$ is initialized as:

$$\Delta\mathbf{G}_v = \{g_{l_{v-1}+1}, \ldots, g_{l_v}\} = \{y^{l_{v-1}+1}, \ldots, y^{l_v}\}, \tag{24}$$

and $|\Delta\mathbf{G}_v| = l'$. Polynomials of $\Delta\mathbf{G}_v$ will then be engaged with the Hasse derivative evaluations and bilinear modifications w.r.t. the constraints of $\Lambda(\mathrm{M}_{v-1})$. Note that the minimal polynomials $f_{(r,s)_{ij}}$ that were identified during the generation of $\widetilde{\mathbf{G}}_{v-1}$ will be reused to update the polynomials of $\Delta\mathbf{G}_v$. After the $\mathcal{C}(\mathrm{M}_{v-1})$ constraints have been satisfied, an updated incremental polynomial group $\Delta\widetilde{\mathbf{G}}_v = \{\tilde{g}_{l_{v-1}+1}, \ldots, \tilde{g}_{l_v}\}$ is obtained. The new polynomials group $\mathbf{G}_v$ is formed by

$$\mathbf{G}_v = \widetilde{\mathbf{G}}_{v-1} \cup \Delta\widetilde{\mathbf{G}}_v. \tag{25}$$

Expansion II: Expand the size (i.e., the number of nonzero coefficients) of the polynomials of group $\mathbf{G}_v$ and enable all the polynomials to satisfy the constraints of $\Lambda(\mathrm{M}_v)$. Since now all polynomials of $\mathbf{G}_v$ satisfy the constraints of $\Lambda(\mathrm{M}_{v-1})$, they only need to perform interpolation w.r.t. the incremental constraints that are defined by $\Lambda(\Delta \mathrm{M}_v)$. Interpolation of this step enables $\mathcal{C}(\mathrm{M}_v) - \mathcal{C}(\mathrm{M}_{v-1})$ extra constraints to be satisfied, resulting in the size of the polynomials being increased. Similarly, we use $\widetilde{\mathbf{G}}_v = \{\tilde{g}_0, \tilde{g}_1, \ldots, \tilde{g}_{l_v}\}$ to denote the updated polynomial group.

After *Expansion II*, the minimal polynomial of $\widetilde{\mathbf{G}}_v$ will be selected to be factorized:

$$Q^{(v)}(x,y) = \min\{\tilde{g}_t \mid \tilde{g}_t \in \widetilde{\mathbf{G}}_v\}. \tag{26}$$

If $u(x) \in L$, decoding will be terminated and outputs $u(x)$. Otherwise, the iteration index will be updated as $v = v + 1$, followed by the OLS update. The next iteration will be performed.

Summarizing the above description, a complete PASD algorithm is stated below.

Algorithm 1 PASD decoding of RS codes

Input: Reliability matrix Π, the maximal OLS l_T and the step size l';
Output: A list of the candidate message polynomials L;
Initialization: Set $\Pi_0^* = \Pi$, $\mathrm{M}_0 = [0]_{q\times n}$, the iteration index $v = 1$, the OLS l_1 and the polynomial group $\mathbf{G}_0 = \{1\}$;
Step 1: Perform the progressive reliability transform to generate matrix M_v;
Step 2: Determine the incremental interpolation constraints $\Lambda(\Delta \mathrm{M}_v)$;
Step 3: Perform *Expansion I* to generate the polynomial group $\mathbf{G}_v$;
Step 4: Perform *Expansion II* to update the group from $\mathbf{G}_v$ to $\widetilde{\mathbf{G}}_v$;
Step 5: Find out the minimal polynomial $Q^{(v)}(x,y)$ from $\widetilde{\mathbf{G}}_v$ and perform factorization. If $u(x) \in L$, terminate the decoding and output $u(x)$; otherwise, go to Step 6;
Step 6: Update the iteration index as $v = v + 1$ and the OLS as (14);
Step 7: If $l_v > l_T$, terminate the decoding and declare a decoding failure; else, go to Step 1;

Notice that when $v = 1$, $\mathrm{M}_0 = [0]_{q\times n}$, and $\Lambda(\mathrm{M}_0) = \emptyset$, *Expansion I* is performed as initializing the polynomial group $\mathbf{G}_1$. Since factorization is performed during each iteration, the number of factorization will be increased compared to the ASD algorithm. However, the computational complexity of the factorization is marginal compared to the interpolation. Increasing the number of factorization will not penalize the system efficiency. This claim will be later justified in Section V. It is important to point out that the proposed algorithm requires extra system memory to store both the minimal polynomials $f_{(r,s)_{ij}}$ and the intermediate matrices Π_v^* and M_v.

More than using the OLS as a decoding constraint, the ASD algorithm can also use the interpolation cost $\mathcal{C}(\mathbf{M})$ or the sum of multiplicities S ($S = \sum_{i,j} m_{ij}$). Hence, either $\mathcal{C}(\mathbf{M})$ or S can be the decoding parameter that is progressively enlarged. In fact, with one of the three parameters being defined, the scales of the other two parameters are also defined. However, the OLS tends to be the most suitable choice. The reason has two folds. First, the ASD algorithm corrects more errors by providing more factorization output candidates, i.e., enlarging the OLS. Second, based on the above description, the OLS needs to be known for interpolation, since it also decides the number of polynomials in the group $\mathbf{G}_v$. Although $\mathcal{C}(\mathbf{M})$ or S maybe chosen as the progressive decoding parameter, the OLS still needs to be known. By using the OLS as a progressive decoding parameter, the algorithm knows when the number of polynomials in the group needs to be increased. It is important to point out that the proposed decoding approach can also be incorporated with the coordinate transform approach of [14] to further facilitate the decoding. Due to the space limit, it is not discussed in this paper, but left as a future work.

IV. Validity Analysis

This section analyzes the validity of the proposed decoding algorithm. Our analysis shows this progressive decoding algorithm only imposes a specific interpolation schedule. Decoding with the same OLS of l_v ($v \leq T$), the progressive interpolation is a natural generalization of the original interpolation problem in the ASD algorithm.

Due to the independence of the interpolation constraints, it is known that for any two distinct constraints $(r_1, s_1)_{i_1 j_1}$ and $(r_2, s_2)_{i_2 j_2}$ that belong to $\Lambda(\mathbf{M}_v)$, one can perform interpolation w.r.t. the constraint $(r_1, s_1)_{i_1 j_1}$, then $(r_2, s_2)_{i_2 j_2}$, or vice versa. That implies interpolation of the ASD algorithm can be carried out with an arbitrary schedule. Its final solution that is seen as the solution for a system with $\mathcal{C}(\mathbf{M}_v)$ linear constraints remains unchanged. Let w denote the interpolation constraint $(r, s)_{ij} \in \Lambda(\mathbf{M}_v)$, kernel $\mathcal{K}_w$ of the constraint is defined as:

$$\mathcal{K}_w = \{Q \in \mathbb{F}_q[x, y] \mid \mathcal{D}_w(Q) = 0, \ \deg_y Q \leq l_v\}. \quad (27)$$

The cumulative kernel $\overline{\mathcal{K}_w}$ can then be defined as: $\overline{\mathcal{K}_w} = \overline{\mathcal{K}_{w-1}} \cap \mathcal{K}_w = \mathcal{K}_1 \cap \mathcal{K}_2 \cap \cdots \cap \mathcal{K}_{w-1} \cap \mathcal{K}_w$ [12]. Hence, performing interpolation w.r.t. all the constraints defined by $\Lambda(\mathbf{M}_v)$ provides the cumulative kernel $\overline{\mathcal{K}_{\mathcal{C}(\mathbf{M}_v)}}$ that is:

$$\begin{aligned}\overline{\mathcal{K}_{\mathcal{C}(\mathbf{M}_v)}} \ = \ & \{Q \in \mathbb{F}_q[x, y] \mid \mathcal{D}_w(Q) = 0, \ \forall \ w \in \Lambda(\mathbf{M}_v) \\ & \text{and } \deg_y Q \leq l_v\}. \end{aligned} \quad (28)$$

Polynomial $Q^{(v)}(x, y)$ is the minimal polynomial of the cumulative kernel:

$$Q^{(v)}(x, y) = \min\{Q(x, y) \mid Q(x, y) \in \overline{\mathcal{K}_{\mathcal{C}(\mathbf{M}_v)}}\}. \quad (29)$$

Based on (20), we know $\Lambda(\mathbf{M}_v) = \Lambda(\Delta\mathbf{M}_1) \cup \Lambda(\Delta\mathbf{M}_2) \cup \cdots \cup \Lambda(\Delta\mathbf{M}_v)$. Decoding with the same OLS of l_v, both the ASD and PASD algorithms satisfy the same set of interpolation constraints, providing the same cumulative kernel. Due to the uniqueness of the minimal candidate of the cumulative kernel, both algorithms offer the same solution for $Q^{(v)}(x, y)$. Therefore, performing the progressive interpolation can be seen as performing a conventional interpolation algorithm with the *progressive interpolation schedule* that is defined as the follows. Given a series of interpolation constraint sets $\Lambda(\Delta\mathbf{M}_1)$, $\Lambda(\Delta\mathbf{M}_2)$, ..., $\Lambda(\Delta\mathbf{M}_v)$ ($v \leq T$), the interpolation w.r.t. the constraint $(r_1, s_1)_{i_1 j_1}$ is always performed prior to the interpolation w.r.t. the constraint $(r_2, s_2)_{i_2 j_2}$, if $(r_1, s_1)_{i_1 j_1} \in \Lambda(\Delta\mathbf{M}_{v'})$, $(r_2, s_2)_{i_2 j_2} \in \Lambda(\Delta\mathbf{M}_v)$ and $v' < v \leq T$.

However, more than rescheduling the interpolation order, the progressive algorithm also implies a progressive polynomial group expansion as indicated by (25). We now prove finding the solution of $\Delta\widetilde{\mathbf{G}}_v$ does not change the solution of $\widetilde{\mathbf{G}}_{v-1}$. According to the polynomial updating rules of (23), the outcome of the polynomial updates is defined by the minimal polynomial $f_{(r,s)_{ij}}$. To justify $\Delta\widetilde{\mathbf{G}}_v$ and $\widetilde{\mathbf{G}}_{v-1}$ can be found separately, we have to prove during the generation of $\Delta\widetilde{\mathbf{G}}_v$, the identities of the found polynomials $f_{(r,s)_{ij}}$ remain unchanged.

Lemma 1: For all the polynomials $f_{(r,s)_{ij}}$ with $(r, s)_{ij} \in \Lambda(\mathbf{M}_{v-1})$, we have

$$\text{lod}(f_{(r,s)_{ij}}) < \text{lod}(y^{l_{v-1}+1}). \quad (30)$$

Proof: Given $Q^{(v-1)}$ as the chosen polynomial at iteration $v-1$, it has satisfied $\mathcal{C}(\mathbf{M}_{v-1})$ interpolation constraints. Based on (42) of [11], its leading order satisfies:

$$\text{lod}(Q^{(v-1)}) \leq \mathcal{C}(\mathbf{M}_{v-1}). \quad (31)$$

$Q^{(v-1)}$ was constructed based on the updating rules of (23). Let $\hat{g} \in \mathbb{F}_q[x, y]$ denote the polynomial that was updated to $Q^{(v-1)}$ by either $Q^{(v-1)} = [\hat{g}, f_{(r,s)_{ij}}]_D$ or $Q^{(v-1)} = [x\hat{g}, \hat{g}]_D$. For the former one, $\text{lod}(f_{(r,s)_{ij}}) < \text{lod}(\hat{g}) = \text{lod}(Q^{(v-1)}) \leq \mathcal{C}(\mathbf{M}_{v-1})$. For the latter one, $f_{(r,s)_{ij}} = \hat{g}$ and $\text{lod}(f_{(r,s)_{ij}}) < \text{lod}(Q^{(v-1)}) \leq \mathcal{C}(\mathbf{M}_{v-1})$. Therefore, for any $f_{(r,s)_{ij}}$ with $(r, s)_{ij} \in \Lambda(\mathbf{M}_{v-1})$:

$$\text{lod}(f_{(r,s)_{ij}}) < \mathcal{C}(\mathbf{M}_{v-1}). \quad (32)$$

Based on (18), it is also known that:

$$l_{v-1} = \Big\lfloor \frac{\Delta_{1,k-1}(\mathcal{C}(\mathbf{M}_{v-1}))}{k-1} \Big\rfloor.$$

It implies:

$$\deg_{1,k-1}(y^{l_{v-1}+1}) > \Delta_{1,k-1}(\mathcal{C}(\mathbf{M}_{v-1})), \quad (33)$$

and consequently

$$\text{lod}(y^{l_{v-1}+1}) > \mathcal{C}(\mathbf{M}_{v-1}). \quad (34)$$

Considering both (32) and (34), the inequality of (30) can be realized. ∎

At iteration v of the progressive interpolation, the incremental polynomial group $\Delta\mathbf{G}_v$ is initialized as (24) in *Expansion I*. Each of its polynomials will perform interpolation w.r.t. the constraints of $\Lambda(\mathbf{M}_{v-1})$. Given a certain constraint $(r, s)_{ij} \in \Lambda(\mathbf{M}_{v-1})$, if all polynomials' Hasse derivative evaluations are zero, no update is required. Otherwise, the minimal polynomial $f_{(r,s)_{ij}}$ needs to be identified. Now, there are two possible cases. In case one, there is no $f_{(r,s)_{ij}}$ stored in memory since all polynomials of $\widetilde{\mathbf{G}}_{v-1}$ satisfied the constraint without performing bilinear modification. The

minimal polynomial $f_{(r,s)_{ij}}$ will be chosen from $\Delta\mathbf{G}_v$ and stored in memory. The bilinear modification is performed only involving the polynomials of $\Delta\mathbf{G}_v$. In case two, there is $f_{(r,s)_{ij}}$ stored in memory. Based on Lemma 1, we know the polynomials of $\Delta\mathbf{G}_v$ will not be chosen as $f_{(r,s)_{ij}}$. The minimal polynomials $f_{(r,s)_{ij}}$ in memory will be used to update the polynomials of $\Delta\mathbf{G}_v$. In both cases, the identities of those existing polynomials $f_{(r,s)_{ij}}$ remain unchanged. Consequently, the solution of $\widetilde{\mathbf{G}}_{v-1}$ remains intact. Therefore, the polynomial groups $\widetilde{\mathbf{G}}_{v-1}$ and $\Delta\widetilde{\mathbf{G}}_v$ can be found separately.

Armed with the above knowledge, we can see that if both algorithms decode with the same OLS of l_v, $\Lambda(\mathrm{M}_v)$ constraints are to be satisfied and $\Lambda(\mathrm{M}_v) = \Lambda(\mathbf{M}_{v-1}) \cup \Lambda(\Delta\mathrm{M}_v)$. For interpolation w.r.t. the constraints of $\Lambda(\mathrm{M}_{v-1})$, the progressive interpolation and the original interpolation deliver the same solution for $\mathbf{G}_v$. Then, polynomials of $\mathbf{G}_v$ will interpolate w.r.t. the constraints of $\Lambda(\Delta\mathrm{M}_v)$. Again, both of the algorithms deliver the same solution for $\widetilde{\mathbf{G}}_v$. Therefore, the progressive interpolation is a natural generalization of the original interpolation algorithm, but imposing a progressive interpolation schedule and a progressive polynomial group expansion.

V. Complexity Analysis

This section analyzes the computational complexity of the PASD algorithm and shows the relationship between the average decoding complexity and the channel condition. The average complexity is measured as the average number of finite field arithmetic operations for decoding one codeword frame at a certain channel condition, e.g., the signal-to-noise ratio (SNR) and denoted by ρ.

Let $\mathcal{O}_{\text{PASD}}(l_T)$ denote the average complexity of the PASD algorithm whose designed maximal OLS is l_T, and it can be formulated as:

$$\mathcal{O}_{\text{PASD}}(l_T) = \sum_{v=1}^{T} \mathcal{P}_{l_v}\mathcal{O}_{l_v} + (1 - \sum_{v=1}^{T}\mathcal{P}_{l_v})\mathcal{O}_{l_T}, \tag{35}$$

where $\mathcal{O}_{l_v}$ denotes the complexity of decoding a codeword frame with an OLS of l_v. And $\mathcal{P}_{l_v}$ denotes the probability of the PASD algorithm produces a valid output with an OLS of l_v, but not with any other OLS that is less than l_v. Recall the successful decoding condition of (13), $\mathcal{P}_{l_v}$ can be stated as:

$$\begin{aligned}\mathcal{P}_{l_v} &= \Pr[S_{\mathrm{M}_v}(\bar{c}) > \Delta_{1,k-1}(\mathcal{C}(\mathrm{M}_v)) \text{ and} \\ &\quad S_{\mathrm{M}_{v'}}(\bar{c}) \le \Delta_{1,k-1}(\mathcal{C}(\mathrm{M}_{v'})), \forall\ v' < v].\end{aligned} \tag{36}$$

In (35), the first term $\sum_{v=1}^{T}\mathcal{P}_{l_v}\mathcal{O}_{l_v}$ represents the average complexity of the successful decoding events. Since the probability of the decoding failure is $1 - \sum_{v=1}^{T}\mathcal{P}_{l_v}$ and it can only occur at iteration T, the second term $(1-\sum_{v=1}^{T}\mathcal{P}_{l_v})\mathcal{O}_{l_T}$ represents the average complexity of the decoding failures. We would later show relationship between the probability $\mathcal{P}_{l_v}$ and the quality of received information Π. Consequently, the average complexity $\mathcal{O}_{\text{PASD}}(l_T)$ becomes channel dependent. In order to fully characterize $\mathcal{O}_{\text{PASD}}(l_T)$, we will now analyze $\mathcal{O}_{l_v}$ and $\mathcal{P}_{l_v}$, respectively.

Theorem 2: The complexity of decoding a codeword frame with an OLS of l_v is:

$$\mathcal{O}_{l_v} = O(\mathcal{C}^2(\mathrm{M}_v)(l_v+1)). \tag{37}$$

Proof: The complexity $\mathcal{O}_{l_v}$ consists of $\mathcal{O}_{l_v}^{\text{int}}$ and $\mathcal{O}_{l_v}^{\text{fac}}$ which are the complexities of the interpolation and the factorization, respectively. Based on Section IV, we know that with the same OLS of l_v, the progressive interpolation and the conventional interpolation are equivalent. We can analyze $\mathcal{O}_{l_v}^{\text{int}}$ as that of the conventional interpolation. The chosen interpolated polynomial $Q^{(v)}$ has a leading order of $\text{lod}(Q^{(v)}) \le \mathcal{C}(\mathrm{M}_v)$ [11]. It implies $Q^{(v)}$ has at most $\mathcal{C}(\mathrm{M}_v)+1$ nonzero coefficients which is referred as the size of the polynomial. Performing the Hasse derivative evaluation and the bilinear modification for such a polynomial requires $O(\mathcal{C}(\mathrm{M}_v)+1)$ field operations. Considering interpolation as a process that manipulates l_v+1 polynomials with a similar size and it has $\mathcal{C}(\mathrm{M}_v)$ iterations, its complexity is

$$\mathcal{O}_{l_v}^{\text{int}} = O(\mathcal{C}(\mathrm{M}_v)(\mathcal{C}(\mathrm{M}_v)+1)(l_v+1)) \cong O(\mathcal{C}^2(\mathrm{M}_v)(l_v+1)). \tag{38}$$

Factorization is implemented by the recursive coefficient search (RCS) algorithm [25] [26] [27]. The computation of the recursive process is dominated by the polynomial update step (see Section 4.2 of [11]). Again, its complexity is proportional to the size of the polynomial. Performing one round of RSC determines one coefficient of the output candidate and it requires $O(\mathcal{C}(\mathrm{M}_v)+1)$ field operations. Since one valid output candidate consists of k coefficients. With an OLS of l_v, the RSC algorithm will be called at most kl_v times [25]. Therefore, at iteration v, the factorization complexity is upper bounded by $O((\mathcal{C}(\mathrm{M}_v)+1)kl_v)$. Since the factorization is run at each iteration, its complexity can be determined by:

$$\begin{aligned}\mathcal{O}_{l_v}^{\text{fac}} &= O((\mathcal{C}(\mathrm{M}_1)+1)kl_1 + (\mathcal{C}(\mathrm{M}_2)+1)kl_2 + \cdots \\ &\quad +(\mathcal{C}(\mathrm{M}_v)+1)kl_v) \\ &= O(k\sum_{\eta=1}^{v}(\mathcal{C}(\mathrm{M}_\eta)+1)l_\eta).\end{aligned} \tag{39}$$

Since $\mathcal{O}_{l_v}^{\text{fac}} < O(kv(\mathcal{C}(\mathrm{M}_v)+1)l_v)$ and practically $kv \ll \mathcal{C}(\mathrm{M}_v)$, we know $\mathcal{O}_{l_v}^{\text{fac}} \ll \mathcal{O}_{l_v}^{\text{int}}$. The decoding complexity is still mainly defined by the interpolation and $\mathcal{O}_{l_v} \cong \mathcal{O}_{l_v}^{\text{int}}$. ■

The above analysis shows that although the factorization is performed at each iteration, the decoding complexity is still dominated by the interpolation process. Complexity $\mathcal{O}_{l_v}$ increases with l_v being enlarged and it is quadratic in $\mathcal{C}(\mathrm{M}_v)$. Since $\mathcal{C}(\mathrm{M}_v)$ is also a function of l_v, the following corollary gives a more apparent relationship between $\mathcal{O}_{l_v}$ and l_v.

Corollary 3: When l_v is sufficiently large, the decoding complexity $\mathcal{O}_{l_v}$ becomes:

$$\mathcal{O}_{l_v} = O\Big(\frac{(k-1)^2}{4}(l_v^4 + l_v^5)\Big). \tag{40}$$

Proof: According to Corollary 5 of [23], when l_v is sufficiently large, we have:

$$\Delta_{1,k-1}(\mathcal{C}(\mathrm{M}_v)) \cong \sqrt{2(k-1)\mathcal{C}(\mathrm{M}_v)}. \tag{41}$$

Based on (18), it is known

$$\Delta_{1,k-1}(\mathcal{C}(\mathrm{M}_v)) \cong (k-1)l_v. \tag{42}$$

In conjunction of the above two equations, we can derive

$$\mathcal{C}(\mathrm{M}_v) \cong \frac{(k-1)l_v^2}{2}. \tag{43}$$

By substituting (43) into (37), (40) can be concluded. ■

Corollary 3 indicates the decoding complexity $\mathcal{O}_{l_v}$ increases exponentially with l_v. It also indicates with the same OLS, the decoding complexity increases as the code rate increases. Note that although the decoding complexity expression of (40) does not include the codeword length n, n does play an important role in the calculation of $\mathcal{C}(\mathrm{M}_v)$. When l_v becomes sufficiently large, $\mathcal{C}(\mathrm{M}_v)$ can be approximated by l_v and k. The above analysis shows the computational price of enlarging the OLS value. Therefore, it is desirable to utilize an appropriate OLS value for the decoding.

The most important feature of the PASD algorithm is that it performs decoding with an adaptable OLS, excluding the occasions of performing a successful decoding with an unnecessarily large OLS value. This feature can be realized by analyzing the successful decoding probability $\mathcal{P}_{l_v}$.

By defining

$$\gamma = \frac{k-1}{n} \text{ and } \Phi_{\Pi}(\bar{c}) = \frac{\sqrt{\sum_{i,j} \pi_{ij}^2}}{S_{\Pi}(\bar{c})}, \tag{44}$$

the successful decoding probability $\mathcal{P}_{l_v}$ can be characterized as the follows.

Theorem 4: The successful decoding probability $\mathcal{P}_{l_v}$ can be approximated as:

$$\mathcal{P}_{l_v} \cong \Pr[l_v = \min \mathcal{L}(\Pi) \mid \mathcal{L}(\Pi) = \{l \mid l > l_{th}(\Pi)\}], \tag{45}$$

where

$$l_{th}(\Pi) = \frac{2 + \frac{\sqrt{n\gamma}}{\sqrt{\sum_{i,j} \pi_{ij}^2}}}{2\gamma[1 - \sqrt{k-1}\Phi_{\Pi}(\bar{c})]}. \tag{46}$$

Proof: The nature of the PASD algorithm is to find the minimal OLS value for the decoding. According to equation (36), probability $\mathcal{P}_{l_v}$ can be seen as the probability that l_v is the minimal OLS value that produces $S_{\mathrm{M}_v}(\bar{c}) > \Delta_{1,k-1}(\mathcal{C}(\mathrm{M}_v))$. Therefore, the intuitive question for defining $\mathcal{P}_{l_v}$ is given the received information Π, what is the minimal OLS value to guarantee a successful decoding. Then, $\mathcal{P}_{l_v}$ is the probability of l_v being such a minimal OLS value. Deriving from (13), the inequality (44) of [8] gives a sufficient condition for the successful decoding. After a few algebraic manipulations, we have an OLS threshold that is stated as (46). It is a random variable. Decoding with an l_v value greater than such a threshold, the successful decoding can be guaranteed. Therefore, given a received information Π, one can define the OLS threshold $l_{th}(\Pi)$ and a set of nonnegative integers $\mathcal{L}(\Pi) = \{l \mid l > l_{th}(\Pi)\}$. Decoding with an OLS value picked up from the set $\mathcal{L}(\Pi)$ can always produce the intended message polynomial. $\mathcal{P}_{l_v}$ can be seen as the probability of l_v being the minimal value of the set $\mathcal{L}(\Pi)$. ■

Theorem 4 shows the probability $\mathcal{P}_{l_v}$ is channel dependent. But the OLS threshold $l_{th}(\Pi)$ is a rather loose bound. Practically, the minimal OLS value for a successful decoding is smaller than the threshold. Also, notice that in a badly deteriorated channel (e.g., $\rho \to -\infty$), we have $\pi_{ij} \cong 1/q$, $\forall\, \pi_{ij} \in \Pi$. Consequently, $\sqrt{\sum_{i,j} \pi_{ij}^2} \cong \sqrt{\frac{n}{q}}$, $S_{\Pi}(\bar{c}) \cong \frac{n}{q}$ and $\Phi_{\Pi}(\bar{c}) \cong \sqrt{\frac{q}{n}}$. As a result, $1 - \sqrt{k-1}\Phi_{\Pi}(\bar{c})$ becomes negative and the OLS threshold looses its validity. On the other hand, it turns out to be hard to draw a closed form expression for the $\mathcal{P}_{l_v}$ that can be seen as a function of Π and provide the bounding reference for probability. Alternatively, we attempt to analyze the OLS threshold w.r.t. the channel condition. Such an analytical insight illustrates the relationship between the probability $\mathcal{P}_{l_v}$ and the channel condition.

Observation 5: By refining $\Phi_{\Pi}(\bar{c}) < \frac{1}{\sqrt{k-1}}$, the OLS threshold $l_{th}(\Pi)$ is a decreasing function w.r.t. ρ.

Proof: Based on (46), it can be seen that the validity of $l_{th}(\Pi)$ can be held when $\Phi_{\Pi}(\bar{c}) < \frac{1}{\sqrt{k-1}}$. Together with such a refinement, the above analysis shows when $\rho \to -\infty$,

$$l_{th}^{(1)}(\Pi) = \lim_{\rho \to -\infty} l_{th}(\Pi) = \frac{2 + \sqrt{\gamma q}}{2\gamma \cdot \Theta(\Pi)}, \tag{47}$$

where $\Theta(\Pi) = 1 - \sqrt{k-1}\Phi_{\Pi}(\bar{c})$ is a function of Π and it tends to 0 as $\Phi_{\Pi}(\bar{c})$ approaches $\frac{1}{\sqrt{k-1}}$. On the other end of the spectrum where the channel is sufficiently good (e.g., $\rho \to +\infty$), $\pi_{ij} \cong 1$ if $i = i_j$, and $\pi_{ij} \cong 0$ otherwise. Consequently, $\sqrt{\sum_{i,j} \pi_{ij}^2} \cong \sqrt{n}$, $S_{\Pi}(\bar{c}) \cong n$ and $\Phi_{\Pi}(\bar{c}) \cong \frac{1}{\sqrt{n}}$. Therefore, when $\rho \to +\infty$,

$$l_{th}^{(2)}(\Pi) = \lim_{\rho \to +\infty} l_{th}(\Pi) = \frac{2 + \sqrt{\gamma}}{2\gamma[1 - \sqrt{(k-1)/n}]}. \tag{48}$$

Since $l_{th}^{(2)}(\Pi) < l_{th}^{(1)}(\Pi)$, and the OLS threshold $l_{th}(\Pi)$ is a decreasing function w.r.t. ρ. ■

In conjunction of Theorem 4 and Observation 5, we have the following remark.

Remark 6: By improving the channel condition, the OLS threshold $l_{th}(\Pi)$ is decreased. Probability $\mathcal{P}_{l_v}$ will therefore be in favor of a smaller l_v value. Consequently, the average decoding complexity $\mathcal{O}_{\text{PASD}}(l_T)$ becomes channel dependent. When the channel condition is sufficiently good, $\mathcal{P}_{l_1}$ approaches 1 and $\mathcal{O}_{\text{PASD}}(l_T) \cong \mathcal{O}_{l_1}$. In contrast, as the channel condition deteriorates, the successful decoding probabilities $\mathcal{P}_{l_v}$ become very small, and $\mathcal{O}_{\text{PASD}}(l_T) \cong \mathcal{O}_{l_T}$.

Tables I and II show the simulation results of $\mathcal{P}_{l_v}$ against the channel SNR for the (15, 5) and the (63, 47) RS codes, respectively. They are measured in the AWGN channel using the binary phase shift keying (BPSK) modulation. The CRC-4 code [28] is used for the output validation. It can be seen that with the SNR being improved, the probability $\mathcal{P}_{l_v}$ starts to be in favor of smaller l_v values. Take the (63, 47) RS code as an example. When SNR = 8dB, most of the decoding events are performed with $l_1 = 1$ and $\mathcal{O}_{\text{PASD}}(5) \cong \mathcal{O}_{l_1}$.

Fig.2 shows the simulation results on the average decoding complexity for PASD decoding of the (15, 5) RS code. It can be seen that the average decoding complexity of the ASD algorithm is insensitive to the channel condition. In contrast, the average decoding complexity of the PASD algorithm can be reduced by increasing ρ. It converges to the minimal level of $\mathcal{O}_{l_1}$ at 7dB. Table I shows with $l_T = 5$, 91.56% of the decoding events are performed with an OLS of 1 at 7dB, leveraging the average decoding complexity. Fig.3 further compares the ASD and the PASD algorithms with the hybrid decoding algorithm [21] for the (63, 47) RS code. The hybrid system incorporates either the ASD or the PASD algorithm with the BM algorithm. The ASD (or PASD) algorithm will only be deployed when the BM algorithm fails. It shows that the average decoding complexity can be further reduced

TABLE I
THE STATISTICS OF $\mathcal{P}_{l_v}$ FOR DECODING THE (15, 5) RS CODE WITH $l_T = 5$.

SNR (dB) / $\mathcal{P}_{l_v}$ (%)	2	3	4	5	6	7	8
$\mathcal{P}_{l_1}$	11.08	22.06	37.93	58.77	78.14	91.56	97.74
$\mathcal{P}_{l_2}$	18.36	29.89	35.96	31.80	19.75	8.18	2.25
$\mathcal{P}_{l_3}$	12.78	14.38	10.98	5.07	1.38	0.20	0.01
$\mathcal{P}_{l_4}$	9.16	7.35	4.98	2.07	0.41	0.04	0.00
$\mathcal{P}_{l_5}$	6.01	4.81	2.95	0.79	0.13	0.01	0.00

TABLE II
THE STATISTICS OF $\mathcal{P}_{l_v}$ FOR DECODING THE (63, 47) RS CODE WITH $l_T = 5$.

SNR (dB) / $\mathcal{P}_{l_v}$ (%)	2	3	4	5	6	7	8
$\mathcal{P}_{l_1}$	0.00	0.00	3.71	32.54	78.55	97.13	99.87
$\mathcal{P}_{l_2}$	0.00	1.51	28.47	56.30	21.25	2.87	0.13
$\mathcal{P}_{l_3}$	0.00	1.61	15.92	6.59	0.17	0.00	0.00
$\mathcal{P}_{l_4}$	0.00	1.23	9.68	1.95	0.02	0.00	0.00
$\mathcal{P}_{l_5}$	0.00	1.22	5.54	0.77	0.00	0.00	0.00

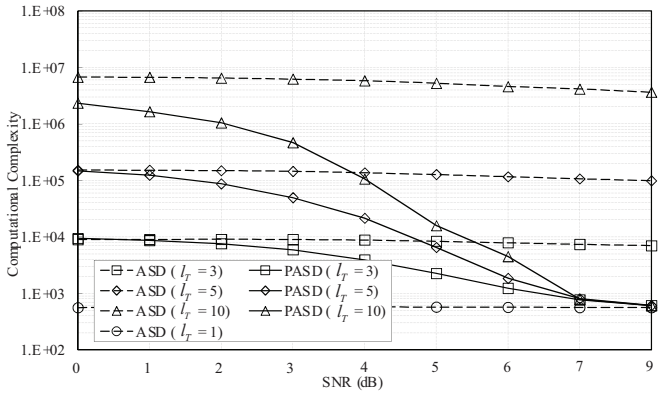

Fig. 2. Average complexity on PASD decoding of the (15, 5) RS code.

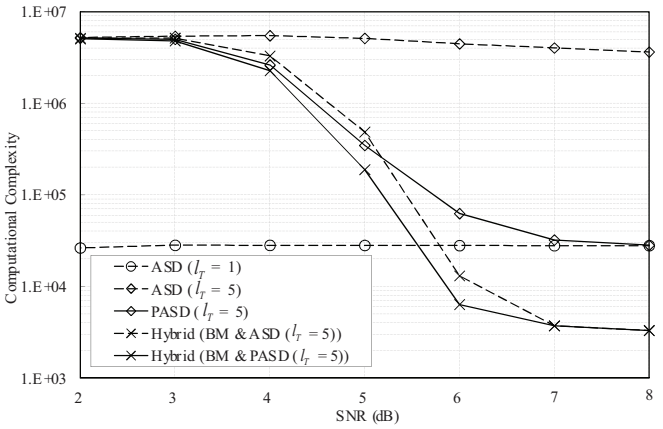

Fig. 3. Average complexity on PASD decoding of the (63, 47) RS code.

and the average decoding complexity of the hybrid system is also channel dependent. With SNR $\leq$ 5.2dB, the average decoding complexity of the PASD algorithm is lower than that of the BM & ASD hybrid system. Notice that for both of the codes, in the low SNR region ($\leq$ 2dB), most of the decoding events reach l_T and perform a large number of factorizations. However, it does not result in a complexity increase compared to the ASD algorithm. The reason is two folds. First, the complexity increase brought by running more factorizations is compensated by the complexity reduction of running the interpolation with a smaller OLS value. Second, in the low SNR region, most of the factorizations do not provide a single output. That says the RSC algorithm has only been called for less than k times. Overall, our results verify Theorem 2 which shows the extra factorization complexity can be neglected. Fig.4 compares the average decoding complexity of the (15, 5) and (15, 11) RS codes. It shows with the same OLS, PASD decoding of the (15, 5) RS code has a lower complexity. It verifies the conclusion of Corollary 3.

As an extra comment to the complexity, the PASD algorithm does require more memory for storing the minimal polynomials $f_{(r,s)_{ij}}$ during the progressive interpolation. Performing the PASD algorithm with an OLS of l_v, the storage complexity is at most $O(\mathcal{C}^2(\mathrm{M}_v))$. Of course, the actual usage of the memory is also channel dependent. With a maximal OLS of l_T, the worst case memory of $O(\mathcal{C}^2(\mathrm{M}_T))$ should be guaranteed.

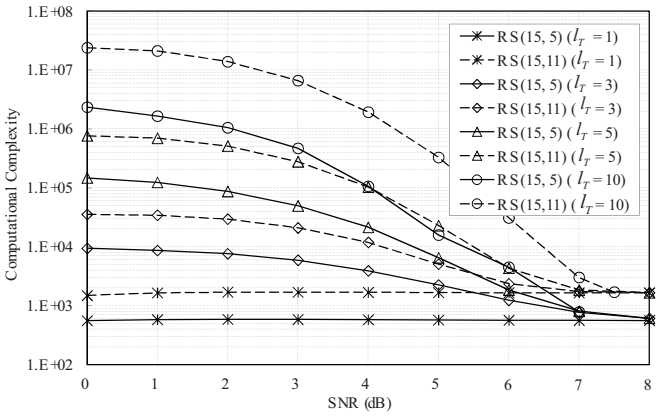

Fig. 4. Average decoding complexity comparison between the (15, 11) and (15, 5) RS codes.

VI. PERFORMANCE EVALUATION

This section presents the error-correction performance of the PASD algorithm. Again, simulations are run in the AWGN

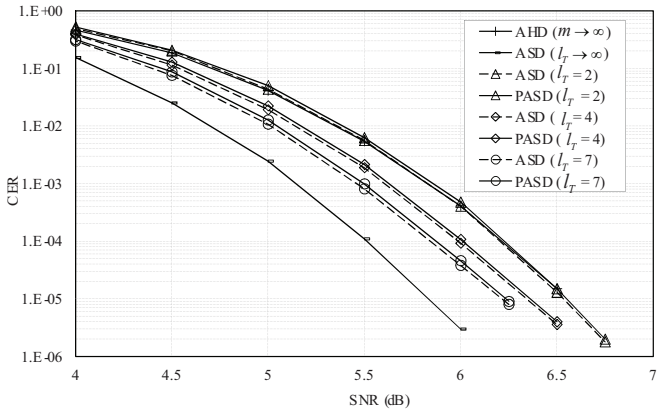

Fig. 5. CER performance of the (63, 47) RS code over the AWGN channel.

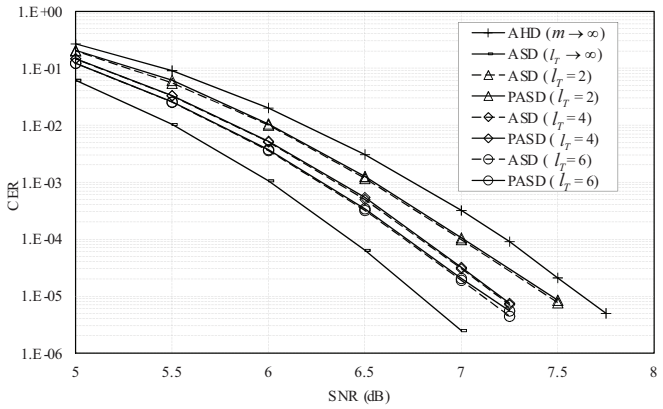

Fig. 6. CER performance of the (63, 55) RS code over the AWGN channel.

channel with the BPSK modulation.

Figs. 5 and 6 show the codeword error rate (CER) of the (63, 47) and (63, 55) RS codes, respectively. The optimal performances of the AHD algorithm (with $m \to \infty$) and the ASD algorithm (with $l_T \to \infty$) are also shown as the comparison benchmarks. The PASD algorithm is to be compared with the ASD algorithm with the same l_T value. For the PASD algorithm, the CRC-4 code is used for output validation. While for the ASD algorithm, the message polynomial is selected using the maximum likelihood (ML) criterion. That means the list of output candidates will be re-encoded, and the one whose codeword has the minimal Euclidean distance to the received vector will be selected. The presented results show the two algorithms perform similarly. Notice that the use of CRC code will cause a loss of code rate. Consequently, a slight performance loss is expected. However, since the CRC code provides a more accurate output message identification than the ML criterion, it compensates the rate loss. Take the (63, 47) RS code as an example. The systems with and without the CRC code have code rates of 0.735 and 0.746, respectively. It is expected that the CRC coded system will suffer $10 \log_{10} \frac{0.746}{0.735} = 0.06$dB performance loss. However, Fig.5 shows the CRC assisted PASD algorithm only suffers at most 0.02dB performance loss at CER of 10^{-5}. In an ideal communication system, one could assume the use of an aided gene. It will notify the decoder once the intended message has been found. In such a system, rate loss is prevented. Fig.7 shows the bit error rate (BER) performance of the widely used (255, 239) RS code with an aided gene and a CRC code. The CRC-16 code is used. The code rates of the gene aided system and the CRC coded system are 0.937 and 0.929, respectively. The CRC coded system suffers 0.03dB performance loss at BER of 10^{-5}. Above all, the presented results demonstrate the practicality of the PASD algorithm. It preserves the error-correction performance of the original ASD algorithm.

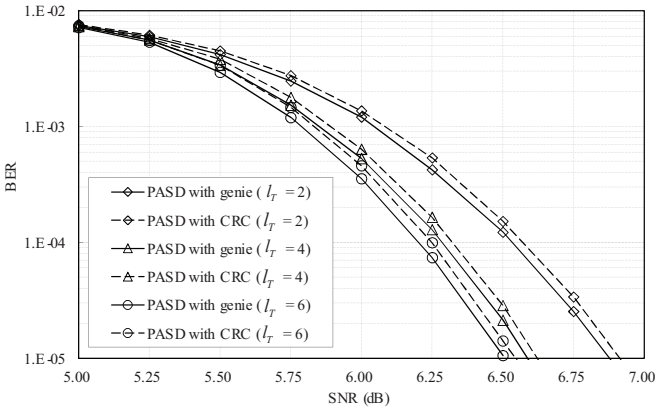

Fig. 7. BER performance of the (255, 239) RS code over the AWGN channel.

VII. Conclusions

This paper has proposed a progressive algebraic soft-decision decoding algorithm for the RS codes. It has been shown that the designed factorization OLS can be enlarged progressively, leading to a progressive interpolation for which the decoding computation can be performed incrementally. More than preserving the error-correction capability, it adjusts the decoding complexity according to the quality of the received information. The validity analysis of the algorithm showed that the progressive interpolation is a natural generalization of the original interpolation problem. With the same decoding OLS, they deliver the same solution. It is important to acknowledge that such a progressive decoding approach is on the expense of the system memory. The complexity analysis further characterized the channel dependence feature of the proposed algorithm. With improving the channel condition, the algorithm will perform more decoding events with a small OLS value and reduce the average decoding complexity. Simulation results reaffirmed the analysis and revealed that in the practically interested SNR region, significant complexity reduction can be achieved. Finally, performance evaluation showed that the PASD algorithm preserves the error-correction performance of the ASD algorithm.

References

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

[2] L. Welch and E. R. Berlekamp, "Error correction for algebraic block codes," in *Proc. 1983 IEEE Int. Symp. Inform. Theory*.

[3] X. Ma and X. M. Wang, "On the minimal interpolation problem and decoding RS codes," *IEEE. Trans. Inf. Theory*, vol. 46, no. 4, pp. 1573–1580, July 2000.

[4] U. Sorger, "A new Reed-Solomon code decoding algorithm based on Newton's interpolation," *IEEE Trans. Inf. Theory*, vol. 39, no. 2, pp. 358–365, Mar. 1993.

[5] R. Koetter, "Fast generalized minimum-distance decoding of algebraic-geometry and Reed-Solomon codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 3, pp. 721–737, May 1996.

[6] M. Sudan, "Decoding of Reed-Solomon codes beyond the error-correction bound," *J. Complexity*, vol. 13, no. 1, pp. 180–193, 1997.

[7] V. Guruswami and M. Sudan, "Improved decoding of Reed-Solomon and algebraic-geometric codes," *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 1757–1767, Sep. 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] A. Vardy and Y. Be'ery, "Bit-level soft-decision decoding of Reed-Solomon codes," *IEEE Trans. Commun.*, vol. 39, no. 3, pp. 440–444, Mar. 1991.

[10] V. Ponnampalam and B. Vucetic, "Soft decision decoding for Reed-Solomon codes," *IEEE Trans. Commun.*, vol. 50, no. 11, pp. 1758–1768, Nov. 2002.

[11] L. Chen, R. A. Carrasco, and E. G. Chester, "Performance of Reed-Solomon codes using the Guruswami-Sudan algorithm with improved interpolation efficiency," *IET Proc. Commun.*, vol. 1, no. 2, pp. 241–250, Apr. 2007.

[12] R. J. McEliece, "The Guruswami-Sudan decoding algorithm for Reed-Solomon codes," *IPN Progress Report*, 42-153, May 2003.

[13] R. Koetter, J. Ma, A. Vardy, and A. Ahmed, "Efficient interpolation and factorization in algebraic soft-decision decoding of Reed-Solomon codes," in *Proc. 2003 IEEE Int. Symp. Inform. Theory*.

[14] 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.

[15] W. J. Gross, F. R. Kschischang, R. Koetter, and P. Gulak, "Towards a VLSI architecture for interpolation-based soft-decision Reed-Solomon decoders," *J. VLSI Signal Process.*, vol. 39, pp. 93–111, Jan.-Feb. 2005.

[16] Y. Wu, "New list decoding algorithms for Reed-Solomon and BCH codes," *IEEE Trans. Inf. Theory*, vol. 54, no. 8, pp. 3611–3630, Aug. 2008.

[17] J. Bellorado and A. Kavcic, "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.

[18] L. Chen, R. A. Carrasco, and M. Johnston, "Reduced complexity interpolation for list decoding of hermitian codes," *IEEE Trans. Wireless Commun.*, vol. 7, no. 11, pp. 4353–4361, Nov. 2008.

[19] Y. Cassuto and J. Bruck, "On the average complexity of Reed-Solomon algebraic list decoders," in *Proc. 2005 IEE/IEEE Int. Symp. on Communication Theory and Applications*.

[20] W. J. Gross, F. R. Kschischang, R. Koetter, and P. Gulak, "Applications of algebraic soft-decision decoding of Reed-Solomon codes," *IEEE Trans. Commun.*, vol. 54, no. 7, pp. 1224–1234, July 2006.

[21] H. Xia, H. Song, and J. R. Cruz, "Retry mode soft Reed-Solomon decoding," *IEEE Trans. Magnetics*, vol. 38, no. 5, pp. 2325–2327, Sep. 2002.

[22] H. Hasse, "Theorie der hoheren Differentiale in einem algebraishen Funcktionenkorper mit vollkommenem konstantenkorper nei beliebeger charakteristic," *J. Reine. Aug. Math.*, pp. 50–54, 1936.

[23] L. Chen, R. A. Carrasco, and M. Johnston, "Soft-decision list decoding of Hermitian codes," *IEEE Trans. Commun.*, vol. 57, no. 8, pp. 2169–2176, Aug. 2009.

[24] R. R. Nielsen, "List decoding of linear block codes," Lyngby, Denmark, Ph.D. thesis, Tech. Univ. Denmark, 2001.

[25] R. 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.

[26] X. W. Wu and P. Siegel, "Efficient root-finding algorithm with application to list decoding of algebraic-geometric codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 6, pp. 2579–2587, Sep. 2001.

[27] L. Chen, R. A. Carrasco, M. Johnston, and E. G. Chester, "Efficient factorization algorithm for list decoding algebraic-geometric and Reed-Solomon codes," in *Proc. 2007 IEEE Int. Conf. Commun.*, pp. 851–856.

[28] T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*. Wiley-Interscience, 2005.

Li Chen (S'07-M'08) received his BSc degree in applied physics from Jinan University, China in 2003, MSc degree in communications and signal processing and PhD degree in mobile communications in 2004 and 2008, respectively, both from Newcastle University of United Kingdom. From 2010, he joined the School of Information Science and Technology, Sun Yat-sen University of China, where he is now an Associate Professor. He is also a Visiting Lecturer with Newcastle University. From 2007 to 2010, he was a Research Associate with Newcastle University. During 2011-12, he is a Visiting Scholar with the Institute of Network Coding, the Chinese University of Hong Kong. He was a recipient of the British Overseas Research Scholarship (ORS). Currently, he is a principle investigator for a National Natural Science Foundation of China (NSFC) project and a co-investigator of the National Basic Research Program (973 program) project. His primary research interests include: information theory, channel coding and wireless communications.

Siyun Tang received her BSc degree in mathematics from Hengyang Normal University, China in 2003, and MSc degree in applied mathematics from Hunan University, China in 2006. She is now a Ph.D. candidate with the School of Information Science and Technology, Sun Yat-sen University, Guangzhou, China. Her research interests include algebraic decoding schemes and their applications in communication systems.

Xiao Ma received his Ph.D. degree in communication and information systems from Xidian University, China, in 2000. From 2000 to 2002, he was a Postdoctoral Fellow with Harvard University, Cambridge, MA. From 2002 to 2004, he was a Research Fellow with City University of Hong Kong. He is now a Professor with the School of Information Science and Technology, Sun Yat-sen University, Guangzhou, China. Prof. Ma is a co-recipient, with A. Kavčić and N. Varnica, of the 2005 IEEE Best Paper Award in Signal Processing and Coding for Data Storage. In 2006, Prof. Ma received the Microsoft Professorship Award from Microsoft Research Asia. Prof. Ma is a member of the IEEE, and his research interests include information theory, channel coding theory and their applications to communication systems and digital recording systems.