Physical Communication 20 (2016) 48–60

Contents lists available at ScienceDirect

Physical Communication

journal homepage: www.elsevier.com/locate/phycom

Full length article

Algebraic soft decoding of Reed–Solomon codes with improved progressive interpolation

Yi Lyu, Li Chen*

School of Information Science and Technology, Sun Yat-sen University, Guangzhou, 510006, China

ARTICLE INFO

Article history:
Received 25 May 2015
Received in revised form 19 October 2015
Accepted 1 June 2016
Available online 10 June 2016

Keywords:
Algebraic soft decoding
Complexity reduction
Koetter–Vardy algorithm
Re-encoding transform
Reed–Solomon codes

ABSTRACT

The algebraic soft decoding (ASD) algorithm for Reed–Solomon (RS) codes can correct errors beyond the half distance bound with a polynomial time complexity. However, the decoding complexity remains high due to the computationally expensive interpolation that is an iterative polynomial construction process. By performing the interpolation progressively, the progressive ASD (PASD) algorithm can adapt the decoding computation to the need, leveraging the average complexity of multiple decoding events. But the complexity reduction is realised at the expense of system memory, since the intermediate interpolation information needs to be memorised. Addressing this challenge, this paper proposes an improved PASD (I-PASD) algorithm that can alleviate the memory requirement and further reduce the decoding complexity. A condition on expanding the set of interpolated polynomials will be introduced, which excepts the need of performing iterative updates for the newly introduced polynomial. Further incorporating the re-encoding transform, the I-PASD algorithm can reduce the decoding complexity over the PASD algorithm by a factor of 1/3 and its memory requirement is at most half of the PASD algorithm. The complexity and memory requirement will be theoretically analysed and validated by numerical results. Finally, we will confirm that the complexity and memory reductions are realised with preserving the error-correction capability of the ASD algorithm.

1. Introduction

Reed–Solomon (RS) codes are widely used in digital communications and storage systems. The conventional unique decoding algorithms for RS codes include Berlekamp–Massey (BM) algorithm [1] and Welch–Berlekamp (WB) algorithm [2,3]. For an (n, k) RS code, where n and k are the length and dimension of the code, respectively, its error-correction capability is limited by $\lfloor \frac{d-1}{2} \rfloor$ where $d = n - k + 1$ is the code's minimum Hamming distance. The algebraic list decoding algorithm [4,5] improves the error-correction capability by a curve-fitting decoding approach, thereby correcting errors beyond the half distance bound. In this paper, it is referred as the algebraic hard decoding (AHD) algorithm. The algebraic soft decoding (ASD) algorithm [6] was later proposed, enhancing the AHD algorithm by introducing an extra process that maps the reliability information to the interpolation multiplicity information. Being able to utilise the soft information provided by the channel, it outperforms the AHD and the unique decoding algorithms.

In algebraic decodings, interpolation that is an iterative polynomial construction process [7–9] dominates the computational complexity and there exists various complexity reduction approaches. In [10], interpolation complexity is reduced by eliminating the interpolated polynomials with a leading order that is greater than the number of interpolation constraints. In [11], a low-complexity Chase (LCC) algebraic decoding algorithm was

* Corresponding author.
E-mail addresses: lvyi3@mail2.sysu.edu.cn (Y. Lyu), chenli55@mail.sysu.edu.cn (L. Chen).

http://dx.doi.org/10.1016/j.phycom.2016.06.001
1874-4907/© 2016 Elsevier B.V. All rights reserved.

proposed. It reduces complexity by exploiting the similarity among the interpolation test-vectors. In [12], by formulating the AHD as a rational curve-fitting problem utilising the outcome of the BM algorithm, i.e., the error locator and error-correction polynomials, it results in a significantly reduced interpolation multiplicity. The re-encoding transform [13–15] is another important approach. Interpolation complexity can be reduced by choosing k received symbols to perform re-encoding, alleviating the iterative polynomial construction computation. Meanwhile, it should be mentioned that there exists other efficient realisations for the interpolation problem, such as the Lee–O'Sullivan approach [16] and its modified variant [17], and the Beelen–Brander approach [18].

The above mentioned approaches were proposed to reduce the computation of a single decoding event. By further observing the fact that different decoding events may require different error-correction capability, the progressive ASD (PASD) [19] algorithm was proposed aiming to reduce the average decoding complexity of multiple decoding events. It functions with a progressively enlarged designed factorisation output list size (OLS), leading to a gradually strengthened error-correction capability. By enlarging the factorisation OLS, both the cardinality of the interpolated polynomial set and the size of each polynomial will be increased. Since such an expansion is realised at the cost of interpolation computation, the PASD algorithm adapts the computation of each individual decoding event to the need, leveraging the average decoding complexity. Other similar efforts include the work of [20] that analyses the interpolation cost's dependence on the received word's error weight. It also proposed an interpolation algorithm that gives priority to update the polynomials that are more likely to be chosen for factorisation. More recently, a multi-trial AHD approach was proposed in [21]. It performs a similar progressive decoding based on the Beelen–Brander interpolation [18].

However, the PASD algorithm's merit in reducing the average decoding complexity is realised at the expense of system memory, since the intermediate interpolation information needs to be memorised. In particular, when a new polynomial is introduced into the set, it needs to be iteratively updated w.r.t. the constraints which have been satisfied by the existing polynomials of the set, during which the intermediate interpolation information is needed. Addressing this challenge, this paper proposes an improved PASD (I-PASD) algorithm that can alleviate the memory requirement and further reduce the decoding complexity. In particular, a condition on expanding the polynomial set will be established such that the newly introduced polynomial is excepted from performing the iterative updates. It further incorporates the re-encoding transform, offering a memory requirement that is at most half of the PASD algorithm and a complexity reduction over the PASD algorithm by a factor of 1/3. Our complexity analysis shows that when the decoding terminates with a factorisation OLS that is greater than one, such a complexity reduction is mainly attributed to the new polynomial set expansion. Both of the complexity and memory analyses of the I-PASD algorithm will be validated by numerical results. Finally, our simulation results confirm that the proposed low complexity algorithm preserves the error-correction capability of the ASD algorithm.

The rest of this paper is organised as the follows. The background knowledge of the paper is presented in Section 2. Section 3 presents I-PASD algorithm. The memory and complexity analyses of the new proposal will be presented in Sections 4 and 5, respectively. The proposed algorithm's error-correction performance will be presented in Section 6. Finally, Section 7 concludes the paper.

2. Background knowledge

2.1. Encoding of RS codes

Let $\mathbb{F}_q = \{\alpha_0, \alpha_1, \ldots, \alpha_{q-1}\}$ denote the finite field of size q, and $\mathbb{F}_q[x]$ and $\mathbb{F}_q[x, y]$ denote the univariate and bivariate polynomial rings defined over $\mathbb{F}_q$, respectively. Given a message vector $\underline{\mu} = (\mu_0, \mu_1, \ldots, \mu_{k-1}) \in \mathbb{F}_q^k$, the message polynomial $\mu(x) \in \mathbb{F}_q[x]$ can be written as:

$$\mu(x) = \mu_0 + \mu_1 x + \cdots + \mu_{k-1}x^{k-1}. \tag{1}$$

A codeword $\underline{c}$ of an (n, k) RS code is generated by:

$$\begin{aligned}\underline{c} &= (c_0, c_1, \ldots, c_{n-1}) \\ &= (\mu(\chi_0), \mu(\chi_1), \ldots, \mu(\chi_{n-1})),\end{aligned} \tag{2}$$

where $\underline{c} \in \mathbb{F}_q^n$. $\chi_0, \chi_1, \ldots, \chi_{n-1}$ are n distinct elements of $\mathbb{F}_q$ and they are called the code locators.

2.2. The ASD algorithm and its progressive variant

It is assumed that an RS codeword is modulated and transmitted through a memoryless channel, e.g., the additive white Gaussian noise (AWGN) channel. Given a received vector $\underline{\mathcal{Y}} \in \mathbb{R}$, the $q \times n$ reliability matrix $\mathbf{\Pi}$ can be obtained, whose entry $\pi_{ij} = \Pr[c_j = \alpha_i \mid \underline{\mathcal{Y}}]$. Matrix $\mathbf{\Pi}$ is then transformed into a multiplicity matrix $\mathbf{M}$ of the same size [6] and its entry m_{ij} represents the interpolation multiplicity for the point (χ_j, α_i), where $\chi_j \in \mathbb{F}_q$ and $c_j = \mu(\chi_j)$. Given a polynomial $Q(x, y) = \sum_{a,b} Q_{ab}x^a y^b \in \mathbb{F}_q[x, y]$ and a nonnegative integer pair (r, s), the (r, s)th Hasse derivative evaluation of Q at point (χ_j, α_i) is defined as [22]:

$$D_{r,s}(Q(x, y))|_{(\chi_j,\alpha_i)} = \sum_{a\geq r, b\geq s} \binom{a}{r}\binom{b}{s} Q_{ab}\, \chi_j^{a-r} \alpha_i^{b-s}. \tag{3}$$

Q interpolates point (χ_j, α_i) with a multiplicity of m_{ij} if $D_{r,s}(Q(x, y))|_{(\chi_j,\alpha_i)} = 0$ for all the (r, s) pairs with $r + s < m_{ij}$. In the following, we will simply use $(r, s)_{ij}$ to denote the interpolation constraint that implies $D_{r,s}(Q(x, y))|_{(\chi_j,\alpha_i)}$. The number of interpolation constraints defined by matrix $\mathbf{M}$ is

$$\mathcal{C}(\mathbf{M}) = \frac{1}{2}\sum_{i=0}^{q-1}\sum_{j=0}^{n-1} m_{ij}(m_{ij} + 1). \tag{4}$$

In decoding an (n, k) RS code, monomials $x^a y^b$ are organised by the $(1, k - 1)$-revlex order.[1] Given a poly-

[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 $a_1 + (k - 1)b_1 < a_2 + (k - 1)b_2$, or $a_1 + (k - 1)b_1 = a_2 + (k - 1)b_2$ and $b_1 < b_2$.

nomial $Q \in \mathbb{F}_q[x, y]$, if $x^{a'}y^{b'}$ is the leading monomial (lm) as $\text{lm}(Q) = x^{a'}y^{b'}$ with coefficient $Q_{a'b'} \neq 0$, 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 (lod) is defined as: $\text{lod}(Q) = \text{ord}(x^{a'}y^{b'})$. Given two polynomials $(H, Q) \in \mathbb{F}_q[x, y]$, we declare $H < Q$ if $\text{lod}(H) < \text{lod}(Q)$. Interpolation is to find a polynomial Q that has the minimal $(1, k-1)$-weighted degree and satisfies all the $\mathcal{C}(\mathbf{M})$ constraints.

Given a multiplicity matrix $\mathbf{M}$, we use i_j to denote the row index i that yields $\alpha_i = c_j$. The codeword score is defined as:

$$S_{\mathbf{M}}(\underline{c}) = \sum_{j=0}^{n-1} m_{i_j j}, \tag{5}$$

and the following successful ASD condition can be introduced.

Theorem 1. *Given $Q \in \mathbb{F}_q[x, y]$ as an interpolated polynomial, if*

$$S_{\mathbf{M}}(\underline{c}) > \deg_{1,k-1}Q, \tag{6}$$

the message polynomial $\mu(x)$ can be found out by the factorising Q as $Q(x, \mu(x)) = 0$ [6].

Proof. For the proof, it is important to be armed with the following prerequisite. If the polynomial $Q \in \mathbb{F}_q[x, y]$ passes through a point (χ_j, α_i) with a multiplicity of m_{ij}, and $p(x) \in \mathbb{F}_q[x]$ is a polynomial that evaluates $p(\chi_j) = \alpha_i$, then $(x - \chi_j)^{m_{ij}} | Q(x, p(x))$. For the message polynomial $u(x)$, we have $u(\chi_j) = c_j$, $\forall j$. Let $Q'(x) = Q(x, \mu(x))$, then

$$(x - \chi_0)^{m_{i_0 0}}(x - \chi_1)^{m_{i_1 1}} \cdots (x - \chi_{n-1})^{m_{i_{n-1} n-1}} \mid Q'(x).$$

The number of x-roots that polynomial Q' has is at least $S_{\mathbf{M}}(\underline{c})$. Since $\deg_x \mu \leq k-1$, $\deg_x Q' \leq \deg_{1,k-1}Q$. Therefore, if $S_{\mathbf{M}}(\underline{c}) > \deg_{1,k-1}Q$, we must have $Q'(x) = 0$. ■

The factorisation [23–25] of Q delivers a list $\mathcal{L}$ of message polynomial candidates that are in the form of $\mu(x)$. The cardinality of $\mathcal{L}$ is referred as the factorisation OLS. Since $\deg_{1,k-1}Q$ is upper bounded by $\Delta_{1,k-1}(\mathcal{C}(\mathbf{M}))$, where $\Delta_{1,k-1}(\mathcal{C}(\mathbf{M})) = \deg_{1,k-1}(x^a y^b | \text{ord}(x^a y^b) = \mathcal{C}(\mathbf{M}))$, the following corollary on the successful decoding condition can be led to.

Corollary 2. *Given an interpolated polynomial $Q \in \mathbb{F}_q[x, y]$, the message polynomial $\mu(x)$ can be found out if*

$$S_{\mathbf{M}}(\underline{c}) > \Delta_{1,k-1}(\mathcal{C}(\mathbf{M})). \tag{7}$$

Proof. It is known that $\deg_{1,k-1}Q \leq \Delta_{1,k-1}(\mathcal{C}(\mathbf{M}))$. If $S_{\mathbf{M}}(\underline{c}) > \Delta_{1,k-1}(\mathcal{C}(\mathbf{M}))$, then $S_{\mathbf{M}}(\underline{c}) > \deg_{1,k-1}Q$. The conclusion follows based on Theorem 1. ■

In order to explain the evolution from the ASD algorithm to its progressive variant, i.e., the PASD algorithm, the following definitions need to be introduced.

Definition 1. Let $\Lambda(m_{ij})$ denote the set of interpolation constraints defined by m_{ij} as: $\Lambda(m_{ij}) = \{(r, s)_{ij}, \forall\, r + s < m_{ij}\}$. $\Lambda(\mathbf{M})$ is used to denote a collection of all the constraint sets $\Lambda(m_{ij})$ that are defined by the nonzero entries of $\mathbf{M}$ as:

$$\Lambda(\mathbf{M}) = \{(r, s)_{ij}, \forall\, m_{ij} \in \mathbf{M} \text{ and } m_{ij} \neq 0\}. \tag{8}$$

Therefore, we have $|\Lambda(\mathbf{M})| = \mathcal{C}(\mathbf{M})$.

Definition 2. Let $\mathbf{M}_A$ and $\mathbf{M}_B$ denote two multiplicity matrices of the same size with entries $m_{ij}^{(A)}$ and $m_{ij}^{(B)}$, respectively. With $m_{ij}^{(A)} \leq m_{ij}^{(B)}$ for all entries, the incremental interpolation constraints introduced by the two matrices are defined as:

$$\begin{aligned}\Lambda(\mathbf{M}_{B\setminus A}) &= \{\Lambda(\mathbf{M}_B) \setminus \Lambda(\mathbf{M}_A)\} \\ &= \{\Lambda(m_{ij}^{(B)}) \setminus \Lambda(m_{ij}^{(A)}), \\ &\qquad \forall\, m_{ij}^{(A)} \in \mathbf{M}_A \text{ and } m_{ij}^{(B)} \in \mathbf{M}_B\}.\end{aligned} \tag{9}$$

Note that $|\Lambda(\mathbf{M}_{B\setminus A})| = \mathcal{C}(\mathbf{M}_B) - \mathcal{C}(\mathbf{M}_A)$.

The PASD algorithm performs decoding with a series of monotonically increasing designed factorisation OLS values $l_1, l_2, \ldots, l_{v-1}, l_v, \ldots, l_T$, where l_T is the maximal OLS set according to the decoding's computational budget. Based on the above OLS series, a corresponding series of multiplicity matrices can be generated as: $\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_{v-1}, \mathbf{M}_v, \ldots, \mathbf{M}_T$, where $m_{ij}^{(v-1)} \leq m_{ij}^{(v)}$. With Definition 2, it can be realised that

$$\Lambda(\mathbf{M}_v) = \Lambda(\mathbf{M}_{1\setminus 0}) \cup \Lambda(\mathbf{M}_{2\setminus 1}) \cup \cdots \cup \Lambda(\mathbf{M}_{v\setminus v-1}), \tag{10}$$

for $v = 1, 2, \ldots, T$. Since $\mathbf{M}_0 = [0]_{q\times n}$ and $\Lambda(\mathbf{M}_0) = \emptyset$, we have $\Lambda(\mathbf{M}_{1\setminus 0}) = \Lambda(\mathbf{M}_1)$. The PASD algorithm performs interpolation w.r.t. the constraints of $\Lambda(\mathbf{M}_1)$, $\Lambda(\mathbf{M}_{2\setminus 1}), \ldots, \Lambda(\mathbf{M}_{v\setminus v-1}), \ldots, \Lambda(\mathbf{M}_{T\setminus T-1})$ progressively, and it will be terminated if the message can be found from its intermediate interpolation outcome.[2] According to Corollary 2, once

$$S_{\mathbf{M}_v}(\underline{c}) > \Delta_{1,k-1}(\mathcal{C}(\mathbf{M}_v)), \tag{11}$$

the decoding will be terminated. However, if interpolation w.r.t. constraints of $\Lambda(\mathbf{M}_{T\setminus T-1})$ has been performed and the intended message polynomial still cannot be found, the decoding will also be terminated and a decoding failure will be declared. Therefore, with a mildly corrupted received information, condition of (11) will happen in an earlier decoding stage with a smaller OLS value. It results in a lower decoding complexity, and vice versa.

3. The improved progressive interpolation

The improved progressive interpolation is characterised by two new features. First, the re-encoding transform will be performed based on matrix $\mathbf{M}_1$, reducing the iterative polynomial construction task of k interpolation points. Second, a polynomial set expanding condition as well as the updating operation for the newly introduced polynomial will be established. Consequently, the newly

[2] The decoding output can be validated by the maximum likelihood (ML) criterion of [26].

introduced polynomial does not need to perform the iterative updates w.r.t. the previous constraints. It cannot only reduce the memory requirement in realising the progressive interpolation, but also reduce the decoding complexity. Based on the above new features, the I-PASD algorithm will be proposed.

3.1. The re-encoding transform based on $\mathbf{M}_1$

The set of interpolation points indicated by matrix $\mathbf{M}_1$ can be defined as:

$$P_{\mathbf{M}_1} = \{(\chi_j, \alpha_i),\ \forall\, m_{ij} \in \mathbf{M}_1 \text{ and } m_{ij} \neq 0\}. \tag{12}$$

We first identify k points in the set $P_{\mathbf{M}_1}$ to perform re-encoding [15]. In matrix $\mathbf{\Pi}$, the maximal entry of each column can be identified as $\pi_j^* = \max\{\pi_{ij} \mid i = 0, 1, \ldots, q-1\}$. Sorting the n maximal entries in a descending order yields a refreshed column index sequence $\theta_0, \theta_1, \ldots, \theta_{k-1}, \ldots, \theta_{n-1}$, which implies $\pi^*_{\theta_0} \geq \pi^*_{\theta_1} \geq \cdots \geq \pi^*_{\theta_{k-1}} \geq \cdots \geq \pi^*_{\theta_{n-1}}$. Let $i(\theta)$ denote the row index of entry π^*_θ as $i(\theta) = \{i \mid \pi_{i,\theta} = \pi^*_\theta\}$, the following set of points can be constituted for re-encoding as:

$$P^{\mathrm{I}}_{\mathbf{M}_1} = \{(\chi_{\theta_0}, \alpha_{i(\theta_0)}), (\chi_{\theta_1}, \alpha_{i(\theta_1)}), \ldots, (\chi_{\theta_{k-1}}, \alpha_{i(\theta_{k-1})})\} \tag{13}$$

and $|P^{\mathrm{I}}_{\mathbf{M}_1}| = k$. Let $\Theta = \{\theta_0, \theta_1, \ldots, \theta_{k-1}\}$, the re-encoding polynomial $T(x)$ can be defined as:

$$T(x) = \sum_{j \in \Theta} \alpha_{i(j)} t_j(x), \tag{14}$$

where $t_j(x)$ is the Lagrange basis polynomial that is written as:

$$t_j(x) = \prod_{\substack{j,\delta \in \Theta,\\ \delta \neq j}} \frac{x - \chi_\delta}{\chi_j - \chi_\delta}. \tag{15}$$

Note that $T(\chi_j) = \alpha_{i(j)},\ \forall\, j \in \Theta$. Therefore, for the original interpolation points (χ_j, α_i), we can transform them by $(\chi_j, \alpha_i + T(\chi_j))$. Consequently, set $P_{\mathbf{M}_1}$ is transformed into

$$P'_{\mathbf{M}_1} = \{(\chi_j, \alpha_i + T(\chi_j)),\ \forall\, m_{ij} \in \mathbf{M}_1 \text{ and } m_{ij} \neq 0\}. \tag{16}$$

In particular, $P^{\mathrm{I}}_{\mathbf{M}_1}$ is transformed into

$$P^{\mathrm{I}'}_{\mathbf{M}_1} = \{(\chi_j, 0),\ \forall\, j \in \Theta\}. \tag{17}$$

Note that in matrix $\mathbf{M}_1$, there can be less than k nonzero entries. The above sorting process aims to enable set $P^{\mathrm{I}}_{\mathbf{M}_1}$ (or $P^{\mathrm{I}'}_{\mathbf{M}_1}$) include most of the points that correspond to the nonzero entries of $\mathbf{M}_1$, so that the following process can reduce the interpolation complexity in its best capability.

At the beginning with an initial factorisation OLS $l_1 = 1$, polynomial set $\mathbf{G}_1 = \{g_0, g_1\}$ is initialised by [15]

$$g_u = y^u \prod_{j \in \Theta} (x - \chi_j)^{[m_{i(j)j} - u]^+}, \tag{18}$$

where $u = 0, 1$ and $[m_{i(j)j} - u]^+ = \max\{m_{i(j)j} - u, 0\}$. Let $\Lambda^{\mathrm{I}}(\mathbf{M}_1)$ denote the set of interpolation constraints defined by points of $P^{\mathrm{I}'}_{\mathbf{M}_1}$, polynomials of $\mathbf{G}_1$ have already satisfied those constraints. They will further perform interpolation w.r.t. the remaining constraints of $\{\Lambda(\mathbf{M}_1) \setminus \Lambda^{\mathrm{I}}(\mathbf{M}_1)\}$. Regarding each constraint $(r, s)_{ij} \in \{\Lambda(\mathbf{M}_1) \setminus \Lambda^{\mathrm{I}}(\mathbf{M}_1)\}$, Hasse derivative evaluation of (3) will be performed for each polynomial of $\mathbf{G}_1$. For those polynomials with $D_{(r,s)_{ij}}(g_u) \neq 0$, the minimal one will be selected as:

$$f = \min\{g_u \mid D_{(r,s)_{ij}}(g_u) \neq 0\}, \tag{19}$$

and the bilinear modification will be performed following [6]:

$$g_u = \begin{cases} g_u - \dfrac{D_{(r,s)_{ij}}(g_u)}{D_{(r,s)_{ij}}(f)} f, & \text{if } g_u \neq f, \quad \text{(a)} \\ (x - \chi_j) f, \quad \text{if } g_u = f. & \text{(b)} \end{cases} \tag{20}$$

We define the above bilinear modification as the *polynomial update* such that the updated polynomial satisfies the current constraint $(r, s)_{ij}$. After interpolation w.r.t. constraints of $\{\Lambda(\mathbf{M}_1) \setminus \Lambda^{\mathrm{I}}(\mathbf{M}_1)\}$ has been performed, an updated polynomial set $\tilde{\mathbf{G}}_1 = \{\tilde{g}_0, \tilde{g}_1\}$ will be obtained. The minimal polynomial $Q_1 = \min\{g_u | g_u \in \tilde{\mathbf{G}}_1\}$ will be chosen for factorisation. If $\mu'(x) \in \mathcal{L}$ as it is the factorisation outcome, the intended message polynomial can be further determined by

$$\mu(x) = \mu'(x) + T(x). \tag{21}$$

If the intended message polynomial cannot be found, the OLS will be increased to $l_2 = l_1 + 1$ forcing another stronger algebraic decoding attempt. In the PASD algorithm, cardinality of the set of interpolated polynomials will be subsequently increased by introducing a new polynomial y^{l_2}. The newly introduced polynomial will then perform updates of (20)(a) by engaging with the minimal polynomials f that were identified and memorised when $l_1 = 1$, so that it can satisfy interpolation constraints of $\Lambda(\mathbf{M}_1)$. Therefore, such an updating process requires memory for storing the intermediate interpolation information. Addressing this challenge, the following subsection introduces a condition on expanding the polynomial set without performing the re-interpolation.

3.2. Polynomial set expanding condition

In order to explicitly formulate the polynomial set expanding condition, the following two definitions are necessary to be introduced.

Definition 3. For the σth interpolation constraint $(r, s)_{ij}$, its corresponding kernel $\mathcal{K}_\sigma$ is defined as the set of polynomials of $\mathbb{F}_q[x, y]$ that satisfy the constraint, i.e.,

$$\mathcal{K}_\sigma = \{Q \in \mathbb{F}_q[x, y] \mid D_{(r,s)_{ij}}(Q) = 0\}. \tag{22}$$

The cumulative kernel $\overline{\mathcal{K}}_\sigma$ can be further defined as:

$$\overline{\mathcal{K}}_\sigma = \overline{\mathcal{K}}_{\sigma-1} \cap \mathcal{K}_\sigma = \mathcal{K}_1 \cap \mathcal{K}_2 \cap \cdots \cap \mathcal{K}_\sigma. \tag{23}$$

Note that $\mathcal{K}_0 = \overline{\mathcal{K}}_0 = \{Q \in \mathbb{F}_q[x, y]\}$, since no interpolation constraint has been imposed.

Definition 4. The set of polynomials of $\mathbb{F}_q[x, y]$ whose leading monomial has a y-degree of u is defined as:

$$\mathcal{W}_u = \{Q \in \mathbb{F}_q[x, y] \mid \deg_y \mathrm{lm}(Q) = u\}. \tag{24}$$

In the conventional ASD algorithm [6] that performs decoding with an OLS of l_T, the iterative polynomial construction begins with a set of polynomials $\mathbf{G}^{(0)} = \{g_0^{(0)} = 1, g_1^{(0)} = y, \ldots, g_u^{(0)} = y^u, g_{u+1}^{(0)} = y^{u+1}, \ldots, g_{l_T}^{(0)} = y^{l_T}\}$.[3] Set $\mathbf{G}^{(0)}$ defines the Gröbner basis of cumulative kernel $\overline{\mathcal{K}}_0$ and each of its polynomials $g_u^{(0)} = \min\{\overline{\mathcal{K}}_0 \cap \mathcal{W}_u\}$. After the σth interpolation constraint has been satisfied, the polynomial set evolves to $\mathbf{G}^{(\sigma)} = \{g_0^{(\sigma)}, g_1^{(\sigma)}, \ldots, g_u^{(\sigma)}, g_{u+1}^{(\sigma)}, \ldots, g_{l_T}^{(\sigma)}\}$, where $g_u^{(\sigma)} = \min\{\overline{\mathcal{K}}_\sigma \cap \mathcal{W}_u\}$. It is important to observe that in set $\mathbf{G}^{(0)}$, $g_{u+1}^{(0)}$ is divisible by $g_u^{(0)}$ and such an observation will lead to the following lemma.

Lemma 3. *If* $\text{lod}(g_u^{(\sigma)}) = \text{lod}(g_u^{(0)})$, *then* $g_{u+1}^{(\sigma)}$ *can be computed by* $yg_u^{(\sigma)}$ [27].

Proof. According to the property of cumulative kernel, if $g_u^{(\sigma)} \in \overline{\mathcal{K}}_\sigma$, then $yg_u^{(\sigma)} \in \overline{\mathcal{K}}_\sigma$. Since $\text{lod}(g_u^{(\sigma)}) = \text{lod}(g_u^{(0)})$, then $\text{lod}(g_u^{(\sigma)}(g_{u+1}^{(0)}/g_u^{(0)})) = \text{lod}(g_{u+1}^{(0)})$. With $g_{u+1}^{(0)}$ being the minimal polynomial in $\overline{\mathcal{K}}_0 \cap \mathcal{W}_{u+1}$, $g_u^{(\sigma)}(g_{u+1}^{(0)}/g_u^{(0)})$ will also be the minimal polynomial in $\overline{\mathcal{K}}_\sigma \cap \mathcal{W}_{u+1}$. Knowing that $g_{u+1}^{(0)}/g_u^{(0)} = y$, $g_{u+1}^{(\sigma)}$ can be computed by $yg_u^{(\sigma)}$. ■

Based on the above statement, it can be realised that in polynomial set $\mathbf{G}^{(\sigma)}$ with $|\mathbf{G}^{(\sigma)}| = u + 1$, as far as polynomial $g_u^{(\sigma)}$ is not chosen as the minimal polynomial f as in (19), its update that is conducted by (20)(a) will not lead to its lod being increased. Consequently, the newly introduced polynomial $g_{u+1}^{(\sigma)}$ can always be generated by $g_{u+1}^{(\sigma)} = yg_u^{(\sigma)}$ such that it is the minimal polynomial in $\overline{\mathcal{K}}_\sigma \cap \mathcal{W}_{u+1}$. Therefore, polynomial set $\mathbf{G}^{(\sigma)}$ should be expanded by introducing a new polynomial $g_{u+1}^{(\sigma)}$ when $g_u^{(\sigma)}$ becomes the minimal polynomial, i.e., when $f = g_u^{(\sigma)}$. The following theorem defines the update of polynomial $g_{u+1}^{(\sigma)}$ once it is introduced into set $\mathbf{G}^{(\sigma)}$.

Theorem 4. *In a polynomial set* $\mathbf{G}^{(\sigma)}$ *with* $|\mathbf{G}^{(\sigma)}| = u + 1$, *if* $g_u^{(\sigma)}$ *is the minimal polynomial that does not satisfy the current interpolation constraint* $(r, s)_{ij}$, *i.e.,* $f = g_u^{(\sigma)}$, *a new polynomial* $g_{u+1}^{(\sigma+1)}$ *should be introduced and updated by* [27]

$$g_{u+1}^{(\sigma+1)} = (y - \alpha_i)g_u^{(\sigma)}, \tag{25}$$

such that it satisfies all the interpolation constraints that polynomial $g_u^{(\sigma)}$ *has satisfied and the current one, i.e.,* $(r, s)_{ij}$.

Proof. Based on Lemma 3, we know that the new polynomial $g_{u+1}^{(\sigma)}$ shall be introduced into the set as $g_{u+1}^{(\sigma)} = yg_u^{(\sigma)}$ so that it is the minimal polynomial in $\overline{\mathcal{K}}_\sigma \cap \mathcal{W}_{u+1}$. To further enable polynomial $g_{u+1}^{(\sigma)}$ satisfy the current constraint, update of (20)(a) needs to be performed, i.e.,

$$\begin{aligned} g_{u+1}^{(\sigma+1)} &= g_{u+1}^{(\sigma)} - \frac{D_{(r,s)_{ij}}(g_{u+1}^{(\sigma)})}{D_{(r,s)_{ij}}(g_u^{(\sigma)})} g_u^{(\sigma)} \\ &= yg_u^{(\sigma)} - \frac{\alpha_i D_{(r,s)_{ij}}(g_u^{(\sigma)})}{D_{(r,s)_{ij}}(g_u^{(\sigma)})} g_u^{(\sigma)} \\ &= (y - \alpha_i)g_u^{(\sigma)}. \end{aligned}$$

As a result, $g_{u+1}^{(\sigma+1)}$ is the minimal polynomial in $\overline{\mathcal{K}}_{\sigma+1} \cap \mathcal{W}_{u+1}$. ■

Theorem 4 formulates the polynomial set expanding condition and the updating operation for the newly introduced polynomial. This update does not require the polynomials f that were identified during the interpolation w.r.t. the previous constraints. Therefore, it reduces the memory requirement in storing the polynomials f as in the PASD algorithm. Moreover, since no iterative update is needed for the newly introduced polynomial, it also reduces the computational cost in expanding the polynomial set.

3.3. The I-PASD algorithm

In the PASD algorithm, the polynomial set will expand once the designed factorisation OLS is increased. For the newly proposed I-PASD algorithm, the polynomial set's expansion may not happen simultaneously with the designed OLS. For this reason, during the progressive iteration v, we use l'_v to denote the maximal y-degree of polynomials in set $\mathbf{G}_v$ as:

$$l'_v = \max\{\deg_y g_u \mid g_u \in \mathbf{G}_v\}. \tag{26}$$

It implies the cardinality of set $\mathbf{G}_v$ is $l'_v + 1$. During the iterations, it is always maintained that $1 \le l'_v \le l_v \le l_T$, where l_v is the designed OLS of progressive iteration v. Note that any polynomial with a y-degree greater than the designed OLS l_v will not be chosen as the minimal polynomial of the set $\mathbf{G}_v$ for factorisation, and the updating computation for the polynomial will be redundant. Hence, in set $\mathbf{G}_v$, we need to ensure $l'_v \le l_v$. Moreover, we let l_v progresses with a step size of one, i.e., $l_{v+1} = l_v + 1$.

It is suffice now to apply Theorem 4 in the new progressive interpolation. Without loss of generality, we now describe the I-PASD algorithm as being performed at iteration v with a designed factorisation OLS of l_v, where $2 \le v < T$. At the beginning of iteration v, we have polynomial set

$$\mathbf{G}_v = \{g_0, g_1, \ldots, g_{l'_v}\}. \tag{27}$$

They all satisfy the constraints of $\Lambda(\mathbf{M}_{v-1})$. At the current iteration, they will perform interpolation as in (19)–(20)(b) w.r.t. the constraints of $\Lambda(\mathbf{M}_{v\backslash v-1})$. Once

$$f = g_{l'_v} \tag{28}$$

during the interpolation for a constraint $(r, s)_{ij}$ of $\Lambda(\mathbf{M}_{v\backslash v-1})$, a new polynomial g^* needs to be introduced by

$$g^* = (y - \alpha_i)g_{l'_v}. \tag{29}$$

[3] In the progressive interpolation, there are two types of iterations. One is the iterative polynomial construction and the other is the progressive iteration. In this paper, we use σ to denote the index of a certain constraint $(r, s)_{ij}$ and so for the iterative polynomial construction, and $\mathbf{G}^{(\sigma)}$ denotes the set of polynomials w.r.t. the σth constraint. Alternatively, we use v to denote the progressive iteration index and $\mathbf{G}_v$ denotes the set of polynomials at progressive iteration v.

Since we need to ensure $\text{lod}(g_{l'_v})$ is unchanged when $g_{l'_v}$ is utilised to generate g^*, the update of (29) should be performed before the update of $g_{l'_v}$ itself utilising (20)(b). In order to ensure $l'_v \leq l_v$, we need to determine whether g^* should be included in set $\mathbf{G}_v$ to interpolate the remaining constraints of $\Lambda(\mathrm{M}_{v\setminus v-1})$. The following two cases can be classified for the set expansion.

Case 1.1: If $l'_v < l_v$, polynomial set $\mathbf{G}_v$ needs to be expanded by

$$\mathbf{G}_v = \{g_0, g_1, \ldots, g_{l'_v}\} \cup \{g^*\}. \tag{30}$$

Afterwards, l'_v is increased by one and $\mathbf{G}_v = \{g_0, g_1, \ldots, g_{l'_v-1}, g_{l'_v}\}$. Polynomials of $\mathbf{G}_v$ will perform interpolation w.r.t. the remaining constraints of $\Lambda(\mathbf{M}_{v\setminus v-1})$.

Case 1.2: If $l'_v = l_v$, polynomial set $\mathbf{G}_v$ does not need to be expanded since the maximal y-degree of polynomials of $\mathbf{G}_v$ already reaches l_v and $\deg_y g^* > l_v$. The new polynomial g^* will not be chosen to be factorised after the current progressive interpolation. Instead, it will be stored in memory. Such an operation guarantees $l'_v \leq l_v$ throughout the progressive interpolation. The existing polynomials of $\mathbf{G}_v$ will proceed to perform interpolation w.r.t. the remaining constraints $(r_1, s_1)_{i_1j_1}, (r_2, s_2)_{i_2j_2}, \ldots, (r_{\mathcal{P}}, s_{\mathcal{P}})_{i_{\mathcal{P}}j_{\mathcal{P}}}$ of $\Lambda(\mathbf{M}_{v\setminus v-1})$[4] and the identified minimal polynomials f during this process will be stored with their constraint identity $(r, s)_{ij}$.

In the end of iteration v, we will have an updated polynomial set $\tilde{\mathbf{G}}_v$ as:

$$\tilde{\mathbf{G}}_v = \{\tilde{g}_0, \tilde{g}_1, \ldots, \tilde{g}_{l'_v}\}, \tag{31}$$

and its polynomials satisfy the constraints of $\Lambda(\mathbf{M}_v)$. The minimal polynomial of $\tilde{\mathbf{G}}_v$, i.e.,

$$Q_v = \min\{\tilde{g}_u \mid \tilde{g}_u \in \tilde{\mathbf{G}}_v\} \tag{32}$$

will be chosen for factorisation. After performing the factorisation, the y-roots of Q_v will be obtained as $\mu'(x)$. Perform the message recovery by (21) to find the intended message polynomial $\mu(x)$. If it cannot be found, the designed factorisation OLS will be updated as $l_{v+1} = l_v + 1$ and matrix $\mathbf{M}_{v+1}$ will be generated accordingly.

In the progressive iteration $v + 1$, polynomial set $\mathbf{G}_{v+1}$ inherits information from $\tilde{\mathbf{G}}_v$ and such a process can again be classified into two cases.

Case 2.1: If $l'_v + 1 < l_{v+1}$, it implies either condition of (28) has not occurred or it has occurred as in **Case1.1**. In the latter case, the newly generated polynomial g^* has already been included in the polynomial set $\tilde{\mathbf{G}}_v$. Hence, there is no expansion from $\tilde{\mathbf{G}}_v$ to $\mathbf{G}_{v+1}$, and

$$\mathbf{G}_{v+1} = \tilde{\mathbf{G}}_v. \tag{33}$$

Case 2.2: If $l'_v + 1 = l_{v+1}$, it implies condition of (28) has occurred as in **Case 1.2** and an expansion is needed following

$$\mathbf{G}_{v+1} = \tilde{\mathbf{G}}_v \cup \{g^*\} = \{\tilde{g}_0, \tilde{g}_1, \ldots, \tilde{g}_{l'_v}, g^*\}. \tag{34}$$

Polynomial g^* will then perform interpolation w.r.t. the constraints of $(r_1, s_1)_{i_1j_1}, (r_2, s_2)_{i_2j_2}, \ldots, (r_{\mathcal{L}}, s_{\mathcal{L}})_{i_{\mathcal{L}}j_{\mathcal{L}}}$ by engaging with the respective polynomials f as in (20)(a). For example, if f is the memorised polynomial w.r.t. the constraint of $(r_1, s_1)_{i_1j_1}$, g^* will be updated by $g^* = g^* - \frac{D_{(r_1,s_1)_{i_1j_1}}(g^*)}{D_{(r_1,s_1)_{i_1j_1}}(f)} f$. After performing interpolation w.r.t. the remaining constraints, it results in an updated polynomial $\tilde{g}^*$ and $\mathbf{G}_{v+1}$ becomes

$$\mathbf{G}_{v+1} = \{\tilde{g}_0, \tilde{g}_1, \ldots, \tilde{g}_{l'_v}, \tilde{g}^*\}. \tag{35}$$

Note that during the update of g^*, it is possible that a constraint does not have a corresponding polynomial f being stored in memory. This is because all polynomials of $\mathbf{G}_v$ had satisfied the constraint without performing the update. In such a case, polynomial g^* becomes the minimal polynomial f w.r.t. the constraint and it shall be utilised to generate a new polynomial as in (29) to be stored in the memory. With defining the polynomial set $\mathbf{G}_{v+1}$, the following interpolation can be performed as in iteration v. The decoding will be terminated either when the intended message polynomial is found or the maximal designed factorisation OLS l_T is exceeded.

We summarise the I-PASD algorithm as in Algorithm 1.

The above algorithm sets up the guideline for implementing the I-PASD algorithm, during which there are two subtle details that should be noticed. First, when $v = 1$, one dose not need to perform the iterative polynomial construction w.r.t. all the constraints of $\Lambda(\mathbf{M}_{1\setminus 0}) = \Lambda(\mathbf{M}_1)$. Since the polynomials of set $\mathbf{G}_1$ already satisfy the constraints of $\Lambda^{\mathrm{I}}(\mathbf{M}_1)$, the iterative polynomial construction is only performed w.r.t. the remaining constraints, i.e., $\{\Lambda(\mathbf{M}_1) \setminus \Lambda^{\mathrm{I}}(\mathbf{M}_1)\}$. Second, when $v = 1$, if there is entry $m_{i(j)j} > 1$ for all $j \in \Theta$, $\text{lod}(g_1) > \text{lod}(y)$ when g_1 is initialised as in (18). Consequently, we cannot utilise Theorem 4 to generate a new polynomial once $f = g_1$ for the first set expansion. However, it is often that with $l_1 = 1$, entries of $\mathbf{M}_1$ will not be greater than one and the introduced polynomial set expanding approach can still be applied.

4. Memory requirement analysis

Based on Section 3, we can see that the I-PASD algorithm does not need to store all the intermediate interpolation information thanks to the new polynomial set expansion. Consequently, it requires less memory than the PASD algorithm. For the memory requirement analysis of the I-PASD algorithm, let us consider the worst case scenario as when the interpolation terminates at iteration v with $\mathcal{C}(\mathbf{M}_v)$ constraints having been satisfied, there are $l_v + 1$ polynomials in set $\mathbf{G}_v$ and $\mathcal{C}(\mathbf{M}_v)(l_v + 1)$ polynomial updates. Regarding each constraint, there is a minimal polynomial f being identified. We now seek to analyse the memory requirement for storing the minimal polynomials f by assuming one polynomial coefficient consumes a memory unit.

We first re-characterise the memory requirement of the PASD algorithm as it will be a memory reduction benchmark for the new proposal. In the end of iteration v, the chosen polynomial Q_v has a leading order $\text{lod}(Q_v)$

[4] Note that $\{(r_1, s_1)_{i_1j_1}, (r_2, s_2)_{i_2j_2}, \ldots, (r_{\mathcal{P}}, s_{\mathcal{P}})_{i_{\mathcal{P}}j_{\mathcal{P}}}\} \subset \Lambda(\mathbf{M}_{v\setminus v-1})$.

Algorithm 1 The I-PASD Algorithm

```
Input: The maximal designed OLS $l_T$ and the reliability matrix $\Pi$;
Output: The message polynomial $\mu(x)$ or $\emptyset$;
Initialisations: Let $v = 1$ and $l_1 = 1$;
 1: Generate matrix $\mathbf{M}_v$ based on $l_v$;
 2: If $v = 1$, perform the re-encoding transform based on $\mathbf{M}_1$ and initialise polynomial set $\mathbf{G}_1$ as in (18); Otherwise, if $l'_{v-1} + 1 < l_v$, generate polynomial set $\mathbf{G}_v$ as in (33), else if $l'_{v-1} + 1 = l_v$, generate polynomial set $\mathbf{G}_v$ as in (34)-(35);
 3: For each constraint $(r, s)_{ij}$ of $\Lambda(\mathbf{M}_{v\setminus v-1})$ {
 4:    Determine $D_{(r,s)_{ij}}(g_u)$ for each polynomial of $\mathbf{G}_v$ as in (3), and identify $f$ as in (19);
 5:      If $f = g_{l'_v}$, generate $g^*$ as in (29);
 6:        If $l'_v < l_v$, expand polynomial set $\mathbf{G}_v$ as in (30); Otherwise, set $\mathbf{G}_v$ is left intact and start to memorise the identified polynomial $f$;
 7:    Perform the polynomial update for set $\mathbf{G}_v$ as in (20a)-(20b); }
 8: Determine $Q_v$ as in (32);
 9: Factorise $Q_v$ to retrieve the intended message polynomial $\mu(x)$;
10: If $\mu(x)$ is found, terminate the decoding; Otherwise, update $l_{v+1} = l_v + 1$ and $v = v + 1$;
11: If $l_v > l_T$, terminate the decoding and declare a decoding failure; Otherwise, go to 1.
```

$\leq \mathcal{C}(\mathbf{M}_v)$ and the identified minimal polynomials f will obey $\text{lod}(f) < \mathcal{C}(\mathbf{M}_v)$ [19]. Hence, they have at most $\mathcal{C}(\mathbf{M}_v)$ nonzero coefficients which is referred as the size of the polynomials. In the PASD algorithm, there are at most $\mathcal{C}(\mathbf{M}_v) - \mathcal{C}(\mathbf{M}_{v-1})$ minimal polynomials f being stored at iteration v. Therefore, when it terminates at iteration v, its total memory consumption can be accumulated by

$$\mathcal{S}_{\text{PASD}}(l_v) = \sum_{v'=1}^{v} \big(\mathcal{C}(\mathbf{M}_{v'}) - \mathcal{C}(\mathbf{M}_{v'-1})\big)\mathcal{C}(\mathbf{M}_{v'}). \tag{36}$$

Note that $\mathcal{C}(\mathbf{M}_0) = 0$. According to Corollary 3 of [19], when l_v is sufficiently large,

$$\mathcal{C}(\mathbf{M}_v) \cong \frac{k-1}{2} l_v^2. \tag{37}$$

We can further formulate $\mathcal{S}_{\text{PASD}}(l_v)$ as:

$$\begin{aligned}\mathcal{S}_{\text{PASD}}(l_v) &\cong \sum_{v'=1}^{v}\left(\frac{k-1}{2}l_{v'}^2 - \frac{k-1}{2}l_{v'-1}^2\right)\frac{k-1}{2}l_{v'}^2 \\ &= \frac{(k-1)^2}{2}\sum_{v'=1}^{v}(2l_{v'}-1)\frac{l_{v'}^2}{2} \\ &\cong \frac{(k-1)^2}{2}\sum_{v'=1}^{v} l_{v'}^3. \end{aligned} \tag{38}$$

Since $\sum_{v'=1}^{v} l_{v'}^3 = \frac{l_v^2(l_v+1)^2}{4}$, $\mathcal{S}_{\text{PASD}}(l_v)$ becomes

$$\mathcal{S}_{\text{PASD}}(l_v) \cong \frac{(k-1)^2}{8} l_v^2 (l_v+1)^2. \tag{39}$$

The I-PASD algorithm offers a memory reduction as the minimal polynomials f will only need to be stored in **Case 1.2** as shown in Section 3.3. In order to analyse the memory reduction, the following lemma is introduced.

Lemma 5. *During iteration v, polynomial g_{l_v} of set $\mathbf{G}_v$ can only become the minimal polynomial f after at least*

$$\frac{k-1}{2}(l_v+1)l_v \tag{40}$$

interpolation constraints have been satisfied.

Proof. With $g_{l_v} = y^{l_v}$, we can determine $\deg_{1,k-1} g_{l_v} = (k-1)l_v$. During the iterative polynomial updates, only (20)(b) will lead to the weighted degree of the polynomial increased by one. At the first iteration, g_1 can become the minimal polynomial f after at least $k-1$ interpolation constraints have been satisfied. Similarly, at the second iteration with $l_2 = 2$, polynomial g_2 can only become f after at least $2(k-1) + (k-1)$ constraints have been satisfied. Following a similar accumulation pattern, at iteration v, polynomial g_{l_v} can only become f after at least

$$\sum_{\lambda=1}^{l_v}(k-1)\lambda = \frac{k-1}{2}(l_v+1)l_v$$

interpolation constraints have been satisfied. ■

Hence, at iteration v, the number of polynomials f that need to be stored is equal to the difference between the total number of constraints $\mathcal{C}(\mathbf{M}_v)$ and the number of constraints that have been satisfied before **Case 1.2** happens, i.e.,

$$\Gamma_v = \mathcal{C}(\mathbf{M}_v) - \frac{k-1}{2}(l_v+1)l_v. \tag{41}$$

Armed with this knowledge, the following theorem further characterises the memory requirement of the I-PASD algorithm.

Theorem 6. *If the I-PASD algorithm terminates at iteration v, its memory requirement is upper bounded by*

$$\mathcal{S}_{\text{I-PASD}}(l_v) < \frac{(k-1)^2}{16} l_v^2 (l_v+1)^2. \tag{42}$$

Proof. The above analysis shows at iteration v, the algorithm requires at most $\Gamma_v \mathcal{C}(\mathbf{M}_v)$ memory for storing the minimal polynomials f. Therefore, when the I-PASD algorithm terminates at iteration v, its memory requirement can be accumulated by

$$\mathcal{S}_{\text{I-PASD}}(l_v) = \sum_{v'=1}^{v} \Gamma_{v'}\mathcal{C}(\mathbf{M}_{v'}). \tag{43}$$

Table 1
Average memory requirement in decoding the (63, 47) RS code.

SNR (dB)	1	2	3	4	5	6	7	8
PASD ($l_T = 5$)	1.86×10^5	1.86×10^5	1.86×10^5	1.84×10^5	6.13×10^4	5.20×10^3	1.60×10^3	1.25×10^1
I-PASD ($l_T = 5$)	8.18×10^4	8.08×10^4	8.06×10^4	7.49×10^4	2.13×10^4	1.19×10^3	2.32×10^1	0
PASD ($l_T = 7$)	6.65×10^5	6.55×10^5	6.43×10^5	5.47×10^5	1.84×10^5	1.02×10^4	1.60×10^3	1.25×10^3
I-PASD ($l_T = 7$)	3.12×10^5	3.10×10^5	3.10×10^5	2.35×10^5	9.04×10^4	4.28×10^3	2.32×10^1	0

Table 2
Average memory requirement in decoding the (255, 239) RS code.

SNR (dB)	1	2	3	4	5	6	7	8
PASD ($l_T = 5$)	4.93×10^6	4.93×10^6	4.93×10^6	4.93×10^6	4.08×10^6	2.57×10^5	9.71×10^4	1.88×10^4
I-PASD ($l_T = 5$)	2.35×10^6	2.35×10^6	2.34×10^6	2.29×10^6	2.18×10^6	1.17×10^5	9×10^1	0
PASD ($l_T = 7$)	2.03×10^7	2.03×10^7	2.03×10^7	2.03×10^7	1.92×10^7	5.78×10^5	9.79×10^4	1.88×10^4
I-PASD ($l_T = 7$)	9.30×10^6	9.30×10^6	9.30×10^6	8.97×10^6	8.80×10^6	2.08×10^5	9×10^1	0

When l_v is sufficiently large, $\Delta_{1,k-1}(\mathcal{C}(\mathbf{M}_v))$ can be approximated by [6]

$$\Delta_{1,k-1}(\mathcal{C}(\mathbf{M}_v)) \cong \sqrt{2(k-1)\mathcal{C}(\mathbf{M}_v)}. \tag{44}$$

Based on the Lemma 15 of [6], we know

$$(k-1)l_v \le \Delta_{1,k-1}(\mathcal{C}(\mathbf{M}_v)) < (k-1)(l_v+1). \tag{45}$$

In conjunction of the above two expressions, we can bound $\mathcal{C}(\mathbf{M}_v)$ by

$$\frac{k-1}{2}l_v^2 \le \mathcal{C}(\mathbf{M}_v) < \frac{k-1}{2}(l_v+1)^2. \tag{46}$$

Therefore, based on (43), we have

$$\begin{aligned}\mathcal{S}_{\text{I-PASD}}(l_v) &< \sum_{v'=1}^{v}\left(\frac{k-1}{2}(l_{v'}+1)^2 - \frac{k-1}{2}(l_{v'}+1)l_{v'}\right)\\ &\quad \times \left(\frac{k-1}{2}(l_{v'}+1)^2\right)\\ &\cong \frac{(k-1)^2}{4}\sum_{v'=1}^{v} l_{v'}^3 = \frac{(k-1)^2}{16}l_v^2(l_v+1)^2. \quad \blacksquare\end{aligned}$$

By comparing $\mathcal{S}_{\text{PASD}}(l_v)$ and $\mathcal{S}_{\text{I-PASD}}(l_v)$, we have the following corollary that quantises the memory reduction achieved by the I-PASD algorithm.

Corollary 7. *If both of the progressive decoding algorithms terminate at iteration v, the I-PASD algorithm offers a memory requirement reduction over the PASD algorithm by a factor that is greater than* $1/2$ *as:*

$$\mathcal{S}_{\text{I-PASD}}(l_v) < \frac{1}{2}\mathcal{S}_{\text{PASD}}(l_v). \tag{47}$$

Proof. Based on (42) and the $\mathcal{S}_{\text{PASD}}(l_v)$ characterisation of (39), the conclusion can be straightforwardly led to. $\blacksquare$

Tables 1 and 2 show the numerical results on the average memory requirement for storing the minimal polynomials f in decoding the (63, 47) and the (255, 239) RS codes, respectively. The simulations are conducted in the AWGN channel using the binary phase shift keying (BPSK) modulation. They are measured by running 10 000 decoding events at each signal-to-noise ratio (SNR). This simulation test bed and measurement setup will also be adopted later in the complexity analysis. However, it is important to emphasise that the simulation test bed is chosen to demonstrate the relationship between the memory requirement and complexity of the I-PASD algorithm and the quality of the channel. The I-PASD algorithm's computational flexibility will not be affected by the choice of channel model. The simulation results show the memory requirements of both of the progressive decoding algorithms are channel dependent. As the SNR increases, more decoding events will be terminated at an earlier decoding stage with a small OLS value, resulting in less memory requirement. In particular, when SNR = 8 dB, memory requirement of the I-PASD algorithm is zero. This is because the message polynomial can be found after the first progressive iteration during which polynomial g_1 has never been identified as the minimal polynomial f. Hence, no memory is required. At SNR = 1 dB, most of the decoding events terminate with l_T and the average memory requirement of the I-PASD algorithm can be characterised by $\mathcal{S}_{\text{I-PASD}}(l_T)$. Our numerical results validate Eq. (42) is an accurate upper bound for the I-PASD algorithm's memory requirement. Moreover, they show that the I-PASD algorithm's memory consumption is less than half of the PASD algorithm, which verifies Corollary 7.

It can be realised that the actual memory requirement and decoding complexity relate to the probability of the progressive decoding terminates with a certain OLS value. This probability has been defined by [19] in which its relationship with the channel condition has also been discussed. Therefore, it is not reinvestigated in this paper. As we aim to characterise the I-PASD algorithm's memory and complexity reductions over the PASD algorithm, this paper's analyses are performed to serve this motivation.

5. Computational complexity analysis

This section further analyses the average decoding complexity of the I-PASD algorithm and it is measured as the average number of finite field arithmetic operations in decoding a codeword frame. Compared to the PASD algorithm, the new proposal offers a complexity reduction as a result of the re-encoding transform and the new progressive polynomial set expansion. Again, computational

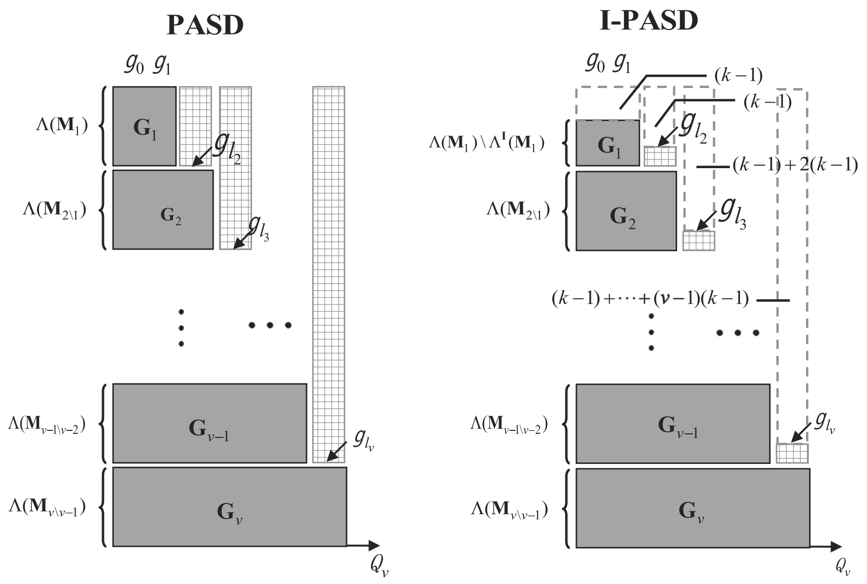
PASD
I-PASD
g_0 g_1
$\Lambda(\mathbf{M}_1)$
$\mathbf{G}_1$
g_{l_2}
$\Lambda(\mathbf{M}_{2\backslash 1})$
$\mathbf{G}_2$
g_{l_3}
$\Lambda(\mathbf{M}_{v-1\backslash v-2})$
$\mathbf{G}_{v-1}$
g_{l_v}
$\Lambda(\mathbf{M}_{v\backslash v-1})$
$\mathbf{G}_v$
Q_v
$(k-1)$
$(k-1)$
$\Lambda(\mathbf{M}_1)\backslash\Lambda^{\mathrm{I}}(\mathbf{M}_1)$
$(k-1)+2(k-1)$
$(k-1)+\cdots+(v-1)(k-1)$

Fig. 1. The polynomial set expansion comparison between the PASD and the I-PASD algorithms.

complexity of the I-PASD algorithm will be derived from analysing the computation reduction over the PASD algorithm. We will first analyse the polynomial update reduction over the PASD algorithm.

For the PASD algorithm that terminates at iteration v, its worst case interpolation complexity can again be analysed by assuming all the $l_v + 1$ polynomials are updated during each of the $\mathcal{C}(\mathbf{M}_v)$ iterations. Consequently, there are $\mathcal{C}(\mathbf{M}_v)(l_v + 1)$ polynomial updates. For the I-PASD algorithm, we also assume that at iteration v the number of polynomials in set $\mathbf{G}_v$ is $l_v + 1$, i.e., $l'_v = l_v$ which also indicates the worst case scenario as in practice $l'_v \le l_v$. The following theorem characterises the polynomial update reduction offered by the I-PASD algorithm.

Theorem 8. *When the I-PASD algorithm terminates at iteration v, it is able to reduce polynomial updates by at most*

$$\Omega(l_v) = \frac{k-1}{6}(l_v^3 - l_v + 6). \tag{48}$$

Proof. Based on Section 4, we know that the $l_v + 1$th ($v \ge 1$) polynomial will be generated and added to set $\mathbf{G}_v$ if g_{l_v} becomes the minimal polynomial f. Based on Lemma 5, we know there are $\frac{k-1}{2}(l_v + 1)l_v$ polynomial updates having been skipped. Therefore, when the I-PASD algorithm terminates at iteration v, it has reduced in total

$$\Xi(l_v) = \sum_{v'=1}^{v-1} \frac{k-1}{2}(l_{v'} + 1)l_{v'} = \frac{k-1}{2}\sum_{v'=1}^{v-1}(l_{v'}^2 + l_{v'}) \tag{49}$$

polynomial updates. With $l_{v+1} = l_v + 1$, we have $l_{v'} = v'$ and $\sum_{v'=1}^{v-1} l_{v'}^2 = \frac{2l_{v-1}^3+3l_{v-1}^2+l_{v-1}}{6}$, $\Xi(l_v)$ can be formulated to

$$\Xi(l_v) = \frac{k-1}{6}(l_v^3 - l_v). \tag{50}$$

Moreover, by performing the re-encoding transform at the beginning of the progressive interpolation, the number of interpolation constraints that were satisfied by the polynomial initialisation is upper bounded by $\mathcal{C}(\mathbf{M}_1)$ as when $\Lambda(\mathbf{M}_1) = \Lambda^{\mathrm{I}}(\mathbf{M}_1)$. Therefore, there are at most $2\mathcal{C}(\mathbf{M}_1)$ polynomial updates having been replaced by the re-encoding transform. Based on (46), it is known that $\mathcal{C}(\mathbf{M}_1) \ge \frac{k-1}{2}$, the number of polynomial updates that are reduced by performing the re-encoding transform will be at most $k-1$. In conjunction with the conclusion of (50), if the I-PASD algorithm terminates at iteration v, it is able to reduce updates by at most

$$\Omega(l_v) = \Xi(l_v) + (k-1) = \frac{k-1}{6}(l_v^3 - l_v + 6). \quad \blacksquare$$

The above proof shows that the I-PASD algorithm reduces the polynomial updates and the decoding complexity over its predecessor due to its two new features, the re-encoding transform and the new polynomial set expansion. It is important to point out that when the progressive decoding terminates with a large OLS, the latter contributes a larger portion in the overall reductions. e.g., with $l_1 = 1$, $\Omega(1) = k-1$ and the polynomial update reduction is mainly due to the re-encoding transform. However, if the decoding terminates with $l_7 = 7$, the new polynomial set expansion will contribute to an update reduction of $56(k-1)$ while the contribution of the re-encoding transform is only $k-1$. This can be better illustrated by Fig. 1 which compares the polynomial set expansion process between the PASD and the I-PASD algorithms with both of the algorithms terminate at iteration v. The progressive polynomial set expansion consists of a horizontal expansion and a vertical expansion, which correspond to introducing a new polynomial into the set and performing updates for the newly introduced polynomial, respectively. The grey areas indicate the existing polynomial set $\mathbf{G}_{v-1}$ performing iterative polynomial construction at iteration $v-1$ to satisfy constraints of $\Lambda(\mathbf{M}_{v-1\backslash v-2})$.

Table 3
Average number of polynomial updates in decoding the (15, 9) RS code.

SNR (dB)	1	2	3	4	5	6	7	8
PASD	2293	2100	1586	780	233	47	20	19
I-PASD	1441	1335	1007	489	141	20	2	1

Table 4
Average number of polynomial updates in decoding the (63, 47) RS code.

SNR (dB)	4	4.5	5	5.5	6	6.5	7	7.5
PASD	11 187	7805	3277	889	229	150	105	96
I-PASD	7 210	5086	2113	549	101	43	9	2

The newly introduced polynomials g_{l_v} are introduced during the expansion as indicated by the pointers. In the PASD algorithm, polynomial g_{l_v} will be introduced into the set $\mathbf{G}_{v-1}$ at the end of iteration $v-1$ and it needs to re-interpolate w.r.t. the previous constraints, i.e., $\Lambda(\mathbf{M}_{v-1})$. While in the I-PASD algorithm, polynomial g_{l_v} will be introduced during iteration $v-1$ and it does not need to perform the re-interpolation. Updates for the newly introduced polynomials are indicated by the grid areas. Therefore, the polynomial update reduction $\Omega(l_v)$ is illustrated by the areas that are bordered by the dash lines in Fig. 1. It can be seen that when the progressive decoding terminates with a large OLS, a large portion of the update reduction is attributed to the newly introduced polynomial set expansion.

The following corollary that further quantises the polynomial update reduction over the PASD algorithm can be straightforwardly derived.

Corollary 9. *When both of the progressive decoding algorithms terminate with a large OLS, the I-PASD algorithm offers a polynomial update reduction over the PASD algorithm by a factor of* 1/3.

Proof. The above analysis shows that when the progressive decodings terminate at iteration v, the PASD algorithm will have performed at most $\mathcal{C}(\mathbf{M}_v)(l_v+1)$ polynomial updates. Based on Theorem 8, we know that the I-PASD algorithm offers a polynomial update reduction over the PASD algorithm by a factor of

$$\begin{aligned}&\frac{(k-1)(l_v^3-l_v+6)}{6}/\mathcal{C}(\mathbf{M}_v)(l_v+1)\\&>\frac{(k-1)(l_v^3-l_v+6)}{6}/\frac{(k-1)(l_v+1)^3}{2}\\&=\frac{1}{3}\cdot\frac{1-l_v^{-2}+6l_v^{-3}}{1+3l_v^{-1}+3l_v^{-2}+l_v^{-3}}.\end{aligned} \tag{51}$$

Therefore, when l_v is large, the polynomial update reduction factor of 1/3 can be led to. ■

Tables 3 and 4 show the numerical results on the average number of polynomial updates in decoding a codeword frame for the (15, 9) and the (63, 47) RS codes, respectively. The maximal designed factorisation OLS is $l_T = 10$. In the low SNR region, most of the decoding events terminate with a large OLS. Our results show that compared to the PASD algorithm, the new proposal reduces the number of polynomial updates by factor that is approximately 1/3, verifying Corollary 9. By increasing the SNR, the progressive decodings will terminate with a small OLS and the average number of polynomial updates decreases. In the high SNR region, most of the decoding events terminate with $l_1 = 1$. For the I-PASD algorithm, the interpolated polynomial Q can often be obtained by the re-encoding transform outcome without performing the iterative polynomial construction. Consequently, its average number of polynomial updates tends to zero.

We can now further characterise the computational complexity of the I-PASD algorithm. Let $\mathcal{O}_{\text{PASD}}(l_v)$ and $\mathcal{O}_{\text{I-PASD}}(l_v)$ denote the computational complexity of the PASD algorithm and the I-PASD algorithm, respectively, with both of the algorithms terminate with an OLS of l_v. Let us recall Corollary 3 of [19] that characterises $\mathcal{O}_{\text{PASD}}(l_v)$ as:

$$\mathcal{O}_{\text{PASD}}(l_v)\cong O\Big(\frac{(k-1)^2}{4}(l_v^5+l_v^4)\Big). \tag{52}$$

The following theorem further characterises $\mathcal{O}_{\text{I-PASD}}(l_v)$ by calculating the computational reduction over $\mathcal{O}_{\text{PASD}}(l_v)$.

Theorem 10. *The computational complexity of the I-PASD algorithm in decoding a codeword frame with an OLS of l_v is*

$$\mathcal{O}_{\text{I-PASD}}(l_v)\cong O\Big(\frac{(k-1)^2}{12}(2l_v^5+3l_v^4)\Big). \tag{53}$$

Proof. Based on Theorem 8, we know with a decoding OLS of l_v, the I-PASD algorithm offers a polynomial update reduction of $\frac{k-1}{6}(l_v^3-l_v+6)$. By assuming each polynomial has at most $\mathcal{C}(\mathbf{M}_v)+1$ coefficients, performing an update for a polynomial needs $O\big(\mathcal{C}(\mathbf{M}_v)+1\big)$ finite field operations, the I-PASD algorithm offers a finite field operation reduction of

$$\Delta\mathcal{O}(l_v)=O\Big(\frac{k-1}{6}(l_v^3-l_v+6)(\mathcal{C}(\mathbf{M}_v)+1)\Big). \tag{54}$$

Based on (37), we know when l_v is sufficiently large, $\mathcal{C}(\mathbf{M}_v)\cong\frac{k-1}{2}l_v^2$ and

$$\begin{aligned}\Delta\mathcal{O}(l_v)&=O\Big(\frac{k-1}{6}(l_v^3-l_v+6)\big(\frac{k-1}{2}l_v^2+1\big)\Big)\\&\cong O\Big(\frac{(k-1)^2}{12}(l_v^5-l_v^3+6l_v^2)\Big).\end{aligned} \tag{55}$$

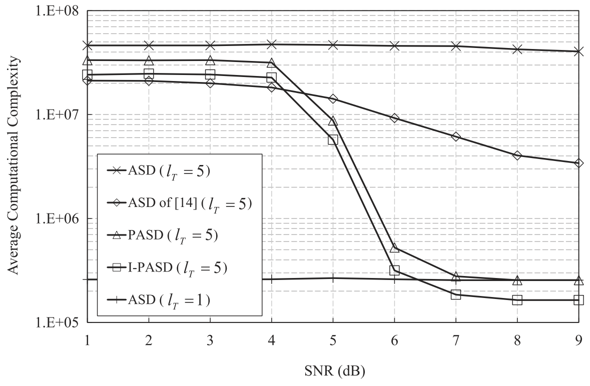

Fig. 2. Average computational complexity in decoding the (63, 47) RS code.

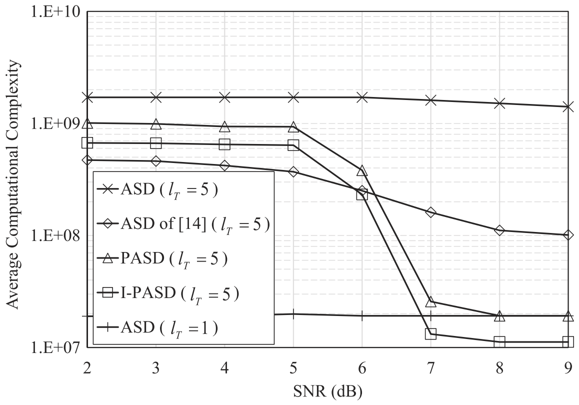

Fig. 3. Average computational complexity in decoding the (255, 239) RS code.

Therefore, $\mathcal{O}_{\text{I-PASD}}(l_v)$ can be determined by:

$$\begin{aligned}\mathcal{O}_{\text{I-PASD}}(l_v) &= \mathcal{O}_{\text{PASD}}(l_v) - \Delta\mathcal{O}(l_v)\\ &\cong O\Big(\frac{(k-1)^2}{12}(2l_v^5 + 3l_v^4 + l_v^3 - 6l_v^2)\Big)\\ &\cong O\Big(\frac{(k-1)^2}{12}(2l_v^5 + 3l_v^4)\Big). \quad \blacksquare\end{aligned}$$

We can see that the computational complexity of the two progressive algorithms increases exponentially with l_v and they are quadratic in the dimension of the code. Therefore, it is desirable to utilise an appropriate OLS value for the decoding, adapting the complexity to the need. Both the PASD and the I-PASD algorithms enable such a function by performing decoding with a progressively enlarged l_v value. Comparing $\mathcal{O}_{\text{I-PASD}}(l_v)$ of (53) and $\mathcal{O}_{\text{PASD}}(l_v)$ of (52) will further lead to the following corollary that quantises the computational complexity reduction offered by the I-PASD algorithm.

Corollary 11. *With the same decoding OLS of l_v, the I-PASD algorithm's computational complexity is approximately* 2/3 *of that of the PASD algorithm.*

Proof. By considering the dominant component of $\mathcal{O}_{\text{PASD}}(l_v)$ and $\mathcal{O}_{\text{I-PASD}}(l_v)$, we have

$$\mathcal{O}_{\text{PASD}}(l_v) \cong O\Big(\frac{(k-1)^2}{8}(2l_v^5 + 2l_v^4)\Big) \cong O\Big(\frac{(k-1)^2}{8}2l_v^5\Big)$$

and

$$\mathcal{O}_{\text{I-PASD}}(l_v) \cong O\Big(\frac{(k-1)^2}{12}2l_v^5\Big).$$

Therefore, $\mathcal{O}_{\text{I-PASD}}(l_v) \cong \frac{2}{3}\mathcal{O}_{\text{PASD}}(l_v)$. $\blacksquare$

In fact, the conclusion of Corollary 11 is in concert with Corollary 9. The computational complexity reduction factor of 1/3 is fundamentally due to the new proposal's capability in reducing the polynomial updates by a factor of 1/3 and this complexity reduction is mainly attributed to the new polynomial set expansion when the progressive decoding terminates with a large factorisation OLS.

Figs. 2 and 3 show the numerical results on the average computational complexity in decoding the (63, 47) and the (255, 239) RS codes, respectively. We compare the I-PASD algorithm with the ASD algorithm, the complexity reducing ASD algorithm of [15] and the PASD algorithm. The two progressive algorithms have the same maximal designed factorisation OLS of $l_T = 5$ which is otherwise a fixed decoding OLS for the ASD algorithm and its complexity reducing variant [15]. As a result, all the algorithms have the same error-correction capability. Note that for the ASD algorithm of [15], its re-encoding transform is performed based on matrix $\mathbf{M}_T$. Our numerical results show that the computational complexity of the I-PASD algorithm is approximately 2/3 of the PASD algorithm, which verifies Corollary 11. Moreover, it can be seen that the average complexity of the ASD algorithm is insensitive to the channel condition. In contrast, the average complexity of the progressive decoding algorithms are channel dependent and it is reduced with an increased SNR. For all the codes, it converges to the minimal level at 8 dB. For the (63, 47) and the (255, 239) RS codes, the ASD algorithm of [15] has a lower complexity than the I-PASD algorithm in the low to medium SNR region. This is because in this region, most of the decoding events produce the intended message vector with a large OLS. The ASD algorithm of [15] is more capable in reducing the complexity by performing the re-encoding transform with matrix $\mathbf{M}_T$ since more iterative polynomial construction can be replaced by polynomial initialisation. However, as the SNR increases, both the PASD and the I-PASD algorithms are far less computationally expensive than the ASD algorithm of [15]. In the high SNR region, most of the progressive decoding events terminate at the first iteration leading to the complexity of the I-PASD algorithm lower than that of the ASD algorithm with $l_T = 1$. Different to the ASD and the PASD algorithms, the I-PASD algorithm further performs the re-encoding transform based on $\mathbf{M}_1$, which benefits a lower decoding complexity. Moreover, it should be pointed out that the progressive decoding algorithms would perform multiple factorisations if the decoding terminates with a factorisation OLS that is greater than one. However, this extra computation can be easily offset by reducing the interpolation complexity which dominates the overall decoding complexity [19]. Finally, it is worthwhile to mention that the proposed algorithm

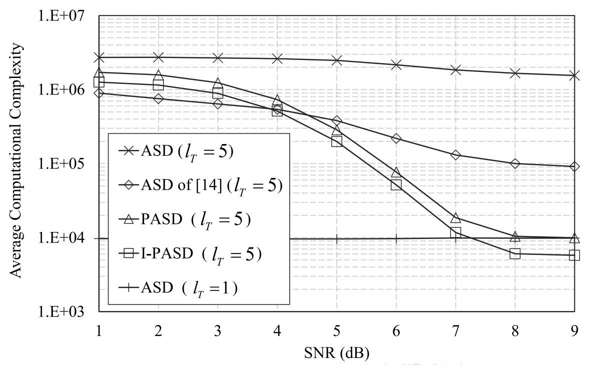

Fig. 4. Average computational complexity in decoding the (15, 13) GRS code.

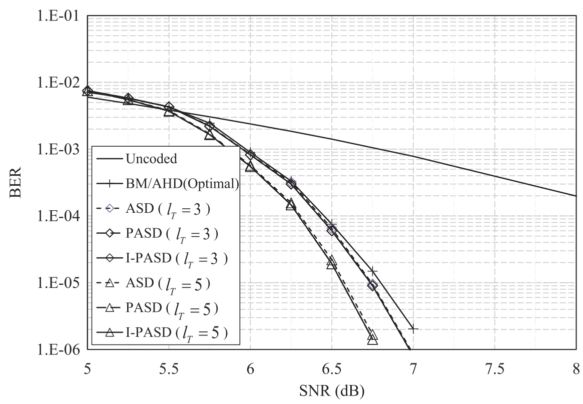

Fig. 5. BER performance of the (255, 239) RS code over the AWGN channel.

can also be applied to decode Generalised RS (GRS) codes, yielding a similar channel dependent feature for its complexity. In order to demonstrate this aspect, Fig. 4 shows the average computational complexity in decoding the (15, 13) GRS code.

6. Error-correction performance

This section presents the error-correction performance of the I-PASD algorithm aiming to confirm the above mentioned memory and complexity reductions are realised without sacrificing the code's performance. Fig. 5 shows the bit error rate (BER) performance of the (255, 239) RS code. All the algebraic soft decoding algorithms, including the PASD, the I-PASD and the ASD algorithms, are functioning with the same maximal OLS value. It shows that all the three algebraic soft decoding algorithms perform similarly, and they have significant performance gains over the BM and AHD algorithms. The optimal AHD results are obtained by assuming that it can correct at most $n - \lfloor\sqrt{n(n-d)}\rfloor - 1$ symbol errors. It can also be noticed that the progressive decoding performance is slightly better than the ASD performance. This marginal performance improvement comes from the fact that multiple factorisations have been performed to find the intended message vector in the progressive decoding mechanism. In Section 2, it has been mentioned that the message polynomial $\mu(x)$ can be found if the successful decoding criterion of (11) is satisfied. There exists some decoding events in which (11) is satisfied with an OLS that is less than l_T. However, when the OLS reaches l_T, (11) is no longer satisfied. Since the ASD algorithm only performs factorisation once with a decoding OLS of l_T, it cannot succeed in decoding the intended message. However, such events rarely happen and consequently the performance improvement delivered by multiple factorisations is marginal. Overall, our simulation results have confirmed that the I-PASD algorithm preserves the error-correction performance of the ASD algorithm.

7. Conclusions

This paper has proposed an improved PASD algorithm for RS codes, reducing the decoding computation and the memory requirement in realising the progressive decoding which is capable of adapting the decoding computation to the need. The improved progressive decoding mechanism is featured by both the re-encoding transform and the new polynomial set expansion. It has been realised that the progressively decoding is at the expense of system memory since the intermediate interpolation information needs to be memorised. Addressing this issue, a polynomial set expanding condition has been established such that the newly introduced polynomial does not need to perform re-interpolation, reducing the memory requirement in storing the intermediate interpolation information. Our memory requirement analysis has shown that the I-PASD algorithm requires less than half of the memory of its predecessor, i.e., the PASD algorithm. The new polynomial set expansion also results in less decoding computation as it has been shown that the number of polynomial updates can be reduced by a factor of 1/3. Further assisted by the re-encoding transform that is performed at the beginning of the progressive decoding, the I-PASD algorithm offers a significant average decoding complexity reduction over the PASD algorithm by a factor of approximately 1/3. Our computational complexity analysis has shown that when the progressive decoding terminates with a large OLS, such a reduction mainly thanks to the new polynomial set expansion. Our numerical results have verified the theoretical analyses and further shown that the I-PASD algorithm is less computationally expensive than various algebraic soft decoding algorithms. Finally, error-correction performance of the I-PASD algorithm has also been presented, showing the memory and complexity reductions are realised with maintaining the ASD performance. Therefore, the proposed algorithm has offered a practical solution for realising the high performance decoding of RS codes. It can be considered to be applied in the RS coded date communication systems.

Acknowledgements

This work is sponsored by the National Natural Science Foundation of China (NSFC) with project ID 61372079, the National Basic Research Program of China (973 Program) with project ID 2012CB316100 and the Fundamental Research Funds for the Central Universities.

References

[1] J.L. Massey, Shift register synthesis and BCH decoding, IEEE Trans. Inform. Theory 15 (1) (1991) 122–127.
[2] L. Welch, E.R. Berlekamp, Error correction for algebraic block codes, in: Proc. IEEE Int. Symp. Inform. Theory, 1983.
[3] P. Gemmell, M. Sudan, Highly resilient correctors for multivariate polynomials, Inform. Process. Lett. 43 (4) (1992) 169–174.
[4] M. Sudan, Decoding of Reed Solomon codes beyond the error-correction bound, J. Complexity 13 (1) (1997) 180–193.
[5] V. Guruswami, M. Sudan, Improved decoding of Reed-Solomon and algebraic-geometric codes, IEEE Trans. Inform. Theory 45 (6) (1999) 1757–1767.
[6] R. Koetter, A. Vardy, Algebraic soft-decision decoding of Reed-Solomon codes, IEEE Trans. Inform. Theory 49 (11) (2003) 2809–2825.
[7] R. Koetter, On algebraic decoding of algebraic-geometric and cyclic codes, (Ph.D. Dissertation), in: Linkoping Studies in Science and Technology, no. 419 Department of Electrical Engineering, Linkoping U., 1996.
[8] R. Koetter, Fast generalized minimum-distance decoding of algebraic-geometric and Reed-Solomon codes, IEEE Trans. Inform. Theory 42 (3) (1996) 721–736.
[9] R. McEliece, The Guruswami-Sudan decoding for Reed-Solomon codes, IPN Progress Report, pp. 42–153, May 2003.
[10] L. Chen, R.A. Carrasco, E.G. Chester, Performance of Reed-Solomon codes using the Guruswami-Sudan algorithm with improved interpolation efficiency, IET Proc. Commun. 1 (2) (2007) 241–250.
[11] J. Bellorado, A. Kavčić, Low-complexity soft-decoding algorithms for Reed-Solomon codes—Part I: An algebraic soft-in hard-out Chase decoder, IEEE Trans. Inform. Theory 56 (3) (2010) 68–79.
[12] Y. Wu, New list decoding algorithm for Reed-Solomon and BCH codes, IEEE Trans. Inform. Theory 54 (8) (2008) 3611–3630.
[13] R. Koetter, J. Ma, A. Vardy, A. Ahmed, Efficient interpolation and factorizaton in algebraic soft-decision decoding of Reed-Solomon codes, in: Proc. IEEE Int. Symp. Inform. Theory, Yokohama, Japan, Jun.-Jul. 2003.
[14] W.J. Gross, F.R. Kschischang, R. Koetter, P. Gulak, Towards a VLSI architecture for interpolation-based soft-decision Reed-Solomon decoders, J. VLSI Signal Process. 39 (2005) 93–111.
[15] R. Koetter, J. Ma, A. Vardy, The re-encoding transformation in algebraic list-decoding of Reed-Solomon codes, IEEE Trans. Inform. Theory 57 (2) (2011) 633–647.
[16] K. Lee, M. O'Sullivan, An interpolation algorithm using Gröbner bases for soft-decision decoding of Reed-Solomon codes, in: Proc. IEEE Int. Symp. Inform. Theory, ISIT, Seattle, USA, Jul. 2006.
[17] P. Trifonov, Efficient interpolation in the Guruswami-Sudan algorithm, IEEE Trans. Inform. Theory 56 (9) (2010) 4341–4349.
[18] P. Beelen, K. Brander, Key equations for list decoding of Reed-Solomon codes and how to solve them, J. Symbolic Comput. 45 (7) (2010) 773–786.
[19] L. Chen, S. Tang, X. Ma, Progressive algebraic soft-decision decoding of Reed-Solomon codes, IEEE Trans. Commun. 61 (2) (2013) 433–442.
[20] Y. Cassuto, J. Bruck, R. McEliece, On the average complexity of Reed-Solomon list decoders, IEEE Trans. Inform. Theory 59 (4) (2013) 2336–2351.
[21] J. Nielsen, A. Zeh, Multi-trial Guruswami-Sudan decoding for generalised Reed-Solomon codes, Des. Codes Cryptogr. (2014) 1–21.
[22] H. Hasse, Theorie der hoheren Differentiale in einem algebraishen Funcktionenkorper mit vollkommenem konstantenkorper nei beliebeger Charakteristic, J. Reine. Aug. Math. (1936) 50–54.
[23] R.M. Roth, G. Ruckenstein, Efficient decoding of Reed-Solomon codes beyond half the minimum distance, IEEE Trans. Inform. Theory 46 (1) (2000) 246–256.
[24] X.W. Wu, P. Siegel, Efficient root-finding algorithm with application to list decoding of algebraic-geometric codes, IEEE Trans. Inform. Theory 47 (6) (2001) 2579–2587.
[25] L. Chen, R.A. Carrasco, M. Johnston, E.G. Chester, Efficient factorisation algorithm for list decoding algebraic-geometric and Reed-Solomon codes, in: Proc IEEE Int. Conf. Commun., Glasgow, UK, May 2007, pp. 851–856.
[26] T. Kaneko, T. Nishijima, H. Inazumi, S. Hirasawa, An efficient maximum-likelihood-decoding algorithm for linear block codes with algebraic decoder, IEEE Trans. Inform. Theory 40 (2) (1994) 320–327.
[27] Y. Zhu, S. Tang, A reduced-complexity algorithm for polynomial interpolation, in: Proc. IEEE Int. Symp. Inform. Theory, ISIT, Jun. 2013, pp. 316–320.

Yi Lyu received his B.Sc. degree in electronic and information engineering from Hunan Normal University in 2012. He is now a master research student with the School of Information Science and Technology, Sun Yat-sen University, China. His research interests include: Reed–Solomon code and its algebraic decoding algorithms.

Li Chen received his B.Sc. degree in applied physics from Jinan University, China in 2003, M.Sc. degree in communications and signal processing and Ph.D. 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 an Associate Head of the Department of Electronic and Communication Engineering. From 2007 to 2010, he was a Research Associate with Newcastle University. During 2011–12, he was a Visiting Scholar with the Institute of Network Coding, the Chinese University of Hong Kong. From Jul. to Sept. 2015, he was a Visiting Researcher with the Institute of Communications Engineering, Ulm University, Germany. Currently, he is a Visiting Associate Professor with Department of Electrical Engineering, University of Notre Dame, United States. He was a recipient of the British Overseas Research Scholarship (ORS). Currently, he is a principle investigator for two National Natural Science Foundation of China (NSFC) projects 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.