# Interpolation-Based Low-Complexity Chase Decoding Algorithms for Hermitian Codes

Siyuan Wu, Li Chen, *Senior Member, IEEE*, and Martin Johnston, *Member, IEEE*

***Abstract*— Algebraic-geometric (AG) codes have good error-correction capability due to their generally large code word length. However, their decoding remains complex, preventing practical applications. Addressing the challenge, this paper proposes two interpolation-based low-complexity Chase (LCC) decoding algorithms for one of the most popular AG codes–Hermitian codes. By choosing $\eta$ unreliable symbols and realizing them with the two most likely decisions, $2^\eta$ decoding test-vectors can be formulated. The first LCC algorithm performs interpolation for the common elements of the test-vectors, producing an intermediate outcome that will be shared by the uncommon element interpolation. It eliminates the redundant computation for decoding each test-vector, resulting in a low-complexity. With an interpolation multiplicity of one, the decoding is further facilitated by removing the requirement of pre-calculating the Hermitian curve's corresponding coefficients. The second LCC algorithm is an adaptive variant of the first algorithm, where the number of test-vectors is determined by the reliability of received information. When the channel condition improves, it can reduce the complexity without compromising the decoding performance. Simulation results show that the both LCC algorithms outperform a number of existing algebraic decoding algorithms for Hermitian codes. Finally, our complexity analysis will reveal the proposals' low-complexity feature.**

***Index Terms*— Algebraic-geometric codes, adaptive decoding, Chase decoding, Hermitian codes, interpolation.**

## I. Introduction

ALGEBRAIC-GEOMETRIC (AG) codes, first introduced by Goppa [1], are a class of linear block codes derived from an algebraic curve. AG codes comprise a large family that include Hermitian codes, Elliptic codes, Reed-Solomon (RS) codes, etc. RS codes are constructed from a straight line and its codeword length cannot exceed the size of finite field, limiting its minimum distance and therefore error-correction capability. In contrast, a general AG code's length can exceed the size of its finite field since more affine points can be found on a curve. This grants AG codes a greater error-correction potential compared to RS codes that are constructed over the same finite field.

For RS and general AG codes, the conventional unique decoding algorithms generate a single decoded message. They utilize syndromes to determine the error locations and error magnitudes. In particular, the well known Berlekamp-Massey (BM) algorithm [2] and the Sakata algorithm [3]–[5] are the unique decoding algorithms for RS and Hermitian codes, respectively. They have a low complexity but cannot correct errors beyond half of the code's minimum Hamming distance $d$, i.e., $\lfloor \frac{d-1}{2} \rfloor$. To correct errors beyond this bound for RS codes, Sudan proposed an interpolation-based list decoding algorithm [6] whose extra error-correction capability only applies to low rate ($< 1/3$) codes. Guruswami and Sudan later generalized the work to list decode both RS and AG codes of all rates, namely the Guruswami-Sudan (GS) algorithm [7]. However, this improved error-correction capability is at the cost of a higher decoding complexity caused by the interpolation. Converting the reliability information into the interpolation information, Koetter and Vardy proposed a soft-decision list decoding algorithm for RS codes, namely the KV algorithm [8]. Soft-decision list decoding of Hermitian codes was later proposed by Chen *et al.* [9] and Lee *et al.* [10], independently. An alternative soft-decision list decoding is the interpolation-based Chase decoding [11] where low complexity can be realized by exploiting the similarity among the decoding test-vectors. This so called low-complexity Chase (LCC) decoding algorithm can outperform various soft-decision decoding algorithms for RS codes at a smaller computational cost.

This paper investigates LCC decoding of Hermitian codes, in which two algorithms will be proposed. The first LCC algorithm extends the RS decoding mechanism [11] to decode Hermitian codes. By choosing $\eta$ unreliable received symbols, we can formulate $2^\eta$ test-vectors in which the unreliable symbols are realized with the two most likely decisions. With such a formulation, the complexity dominant interpolation can be performed for the common elements and the uncommon elements, respectively. The common element interpolation is performed once and its outcome will be shared by the uncommon element interpolation. Exploiting the similarity of the test-vectors, the uncommon element interpolation can be performed in a binary tree expansion manner, eliminating the redundant computation for decoding each test-vector and resulting in a low complexity. Since the LCC decoding

Manuscript received May 8, 2017; revised September 27, 2017 and December 13, 2017; accepted December 15, 2017. Date of publication December 25, 2017; date of current version April 16, 2018. This work is sponsored by the National Natural Science Foundation of China (NSFC) with project IDs 61671486 and 61372079. Part of this work has been published in the 2016 IEEE Information Theory Workshop with article titled "Low-complexity Chase decoding of algebraic-geometric codes using Koetter's interpolation". The associate editor coordinating the review of this paper and approving it for publication was K. Abdel-Ghaffar. *(Corresponding author: Li Chen.)*

S. Wu and L. Chen are with the School of Electronics and Communication Engineering, Sun Yat-sen University, Guangzhou 510006, China (e-mail: wusy7@mail2.sysu.edu.cn; chenli55@mail.sysu.edu.cn).

M. Johnston is with the School of Engineering, Newcastle University, Newcastle upon Tyne NE1 7RU, U.K. (e-mail: martin.johnston@ncl.ac.uk).

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

Digital Object Identifier 10.1109/TCOMM.2017.2786667

performs with an interpolation multiplicity of one, the Hermitian curve's corresponding coefficients do not need to be pre-calculated, which was the key challenge in implementing GS and KV decoding of Hermitian codes [12]. We will show LCC decoding of Hermitian codes yields a good performance and a lower complexity than several existing decoding candidates. It is known that when the channel condition improves, e.g., signal-to-noise ratio (SNR) increases, the received information will be more reliable. Intuitively, less test-vectors will be needed in the Chase decoding for message recovery. Hence, the second LCC algorithm is an adaptive variant of the first algorithm in which the number of test-vectors is altered according to the reliability of the received information. It is realized by setting a reliability threshold for determining the number of unreliable symbols. When the SNR increases, fewer unreliable symbols will be considered resulting in fewer Chase decoding trials and therefore has a reduced message recovering effort. We will show that by carefully choosing the reliability threshold, the adaptive LCC algorithm can maintain the Chase decoding gains with a smaller computational cost. We will also show that both LCC algorithms can outperform a number of existing algebraic decoding algorithms for Hermitian codes. Chase decoding performance for RS and Hermitian codes will also be compared showing the Hermitian codes' error-correction potential. A complexity analysis of the LCC decodings will be carried out to reveal their low-complexity feature.

The rest of this paper is organized as follows. Section II provides the preliminaries of the paper. The two proposed LCC algorithms will be introduced in Sections III and IV, respectively. Section V analyzes the LCC decoding complexity. Section VI shows our simulation results. Finally, Section VII concludes the paper.

## II. Preliminaries

This section presents the preliminaries, including the encoding of Hermitian codes and the interpolation-based list decoding.

### *A. Hermitian Codes*

Let $\mathbb{F}_q = \{\sigma_0, \sigma_1, \ldots, \sigma_{q-1}\}$ denote the finite field of size $q$. $\mathbb{F}_q[x, y]$ and $\mathbb{F}_q[x, y, z]$ denote the bivariate and trivariate polynomial rings over $\mathbb{F}_q$, respectively. An affine Hermitian curve[1] defined over $\mathbb{F}_q$ can be written as [13]

$$H_w(x, y) = x^{w+1} + y^w + y, \tag{1}$$

where $w = \sqrt{q}$ and the curve has a genus $g = \frac{w(w-1)}{2}$. Note that Hermitian codes are constructed over finite fields whose size is a square. There are $w^3$ affine points $p_j = (x_j, y_j)$ that satisfy $H_w(x_j, y_j) = 0$, and a point at infinity $p_\infty$. Let $\mathcal{P}_w$ denote the set of affine points that are found over the curve $H_w$ as $\mathcal{P}_w = \{p_j = (x_j, y_j) \mid H_w(x_j, y_j) = 0\}$, and $|\mathcal{P}_w| = w^3$.

The pole basis $\mathcal{L}_w$ of a Hermitian curve comprises a set of bivariate monomials $\phi_a(x, y) = x^\mu y^\nu$ with increasing pole order at the point of infinity $p_\infty$ as $\mathcal{L}_w = \{\phi_a(x, y) \mid \upsilon_{p_\infty}(\phi_a^{-1}(x, y)) < \upsilon_{p_\infty}(\phi_{a+1}^{-1}(x, y)),\ a \in \mathbb{N}\}$, where $\upsilon_{p_\infty}(\phi_a^{-1}(x, y)) = \upsilon_{p_\infty}((x^\mu y^\nu)^{-1}) = w \cdot \mu + (w + 1) \cdot \nu$ is the pole order of $\phi_a$ and $0 \leq \mu \leq w$, $\nu \geq 0$. For example, over the affine curve $H_2$, $\mathcal{L}_2 = \{1, x, y, x^2, xy, y^2, x^2y, xy^2, y^3, \ldots\}$. In decoding a Hermitian code constructed over $H_w$, polynomials of $\mathbb{F}_q[x, y]$ and $\mathbb{F}_q[x, y, z]$ are constituted with monomials in the basis $\mathcal{L}_w$. For each affine point $p_j$, there exists a zero basis which comprises bivariate polynomials with an increasing zero order over $p_j$. The zero basis polynomials are defined as [14]

$$\begin{aligned}\psi_{p_j,\alpha}(x, y) = (x - x_j)^\lambda[(y - y_j) - x_j^w(x - x_j)]^\delta,\\ (\lambda, \delta) \in \mathbb{N},\end{aligned} \tag{2}$$

where $\alpha = \lambda + (w + 1)\delta$. Polynomial $\psi_{p_j,\alpha}$ has a multiplicity of $\alpha$ at $p_j$.

To construct an $(n, k)$ Hermitian code, where $n$ and $k$ are the length and dimension of the code, respectively, the message polynomial $f(x, y) \in \mathbb{F}_q[x, y]$ is defined as

$$f(x, y) = f_0\phi_0 + f_1\phi_1 + \cdots + f_{k-1}\phi_{k-1}. \tag{3}$$

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

$$\underline{c} = (f(p_0), f(p_1), \ldots, f(p_{n-1})). \tag{4}$$

Since there are $w^3$ affine points $p_j$, the codeword length can reach $w^3$ which exceeds the size of finite field. Hence, $k < n \leq w^3$. The minimum Hamming distance of the code is $d = n - k - g + 1$.

### *B. GS Decoding of Hermitian Codes*

For GS decoding of an $(n, k)$ Hermitian code, the following definitions are needed.

*Definition I:* Trivariate monomials $\phi_a z^b$ are ordered under the $(1, w_z)$-weighted degree

$$\deg_{1,w_z} \phi_a z^b = \upsilon_{p_\infty}(\phi_a^{-1}) + w_z b, \tag{5}$$

where $w_z = \upsilon_{p_\infty}(\phi_{k-1}^{-1})$. Consequently, the $(1, w_z)$-lexicographic order can be established as follows. Given two monomials $\phi_{a_1}z^{b_1}$ and $\phi_{a_2}z^{b_2}$, we claim $\mathrm{ord}(\phi_{a_1}z^{b_1}) < \mathrm{ord}(\phi_{a_2}z^{b_2})$, if $\deg_{1,w_z}\phi_{a_1}z^{b_1} < \deg_{1,w_z}\phi_{a_2}z^{b_2}$, or $\deg_{1,w_z}\phi_{a_1}z^{b_1} = \deg_{1,w_z}\phi_{a_2}z^{b_2}$ and $b_1 < b_2$.

*Definition II:* Given a polynomial $Q(x, y, z) = \sum_{a,b\in\mathbb{N}} Q_{ab}\phi_a(x, y)z^b$, if $\phi_{a'}z^{b'}$ with coefficient $Q_{a'b'} \neq 0$ is the leading monomial, the $(1, w_z)$-weighted degree of $Q$ is $\deg_{1,w_z} Q = \deg_{1,w_z} \phi_{a'}z^{b'}$ and its leading order is $\mathrm{lod}(Q) = \mathrm{ord}(\phi_{a'}z^{b'})$. Given two polynomials $Q_1$ and $Q_2$, we claim $Q_1 < Q_2$, if $\mathrm{lod}(Q_1) < \mathrm{lod}(Q_2)$.

The GS algorithm has two steps, interpolation and factorization. The interpolation builds a polynomial $Q(x, y, z)$ based on the received word $\underline{r} = (r_0, r_1, \ldots, r_{n-1}) \in \mathbb{F}_q^n$. It has a multiplicity of $m$ over points $(p_0, r_0)$, $(p_1, r_1)$, …, $(p_{n-1}, r_{n-1})$ [15]. Polynomial $Q$ will be factorized and its $z$-roots are the decoded message candidates $\hat{f}(x, y)$ [16], [17]. For GS decoding, interpolation dominates the complexity. Hence, reducing the computation for $Q$ is the key to facilitate the decoding.

[1] Equation (1) is an affine component of the projective Hermitian curve $H_w(x, y, z) = x^{w+1} + y^w z + yz^w$.

With the received word $\underline{r}$, the Hamming distance between $\underline{c}$ and $\underline{r}$ is

$$d_{\mathrm{H}}(\underline{c}, \underline{r}) = |\{j | c_j \neq r_j, \forall j\}|. \tag{6}$$

*Theorem 1 [18]:* Given a polynomial $Q \in \mathbb{F}_q[x, y, z]$ that has a zero of multiplicity $m$ over the $n$ points $(p_j, r_j)$, if $m(n - d_{\mathrm{H}}(\underline{c}, \underline{r})) > \deg_{1,w_z} Q$, then $Q(x, y, f) = 0$ or $(z - f) | Q(x, y, z)$.

The interpolation constraint for a polynomial in $\mathbb{F}_q[x, y, z]$ is explained as follows. Given an interpolation point $(p_j, r_j)$, polynomial $Q$ can be written as

$$Q(x, y, z) = \sum_{\alpha,\beta \in \mathbb{N}} Q_{\alpha,\beta}^{(p_j,r_j)} \psi_{p_j,\alpha}(x, y)(z - r_j)^{\beta}, \tag{7}$$

where $Q_{\alpha,\beta}^{(p_j,r_j)} \in \mathbb{F}_q$ and $Q_{\alpha,\beta}^{(p_j,r_j)} = 0$ for $\alpha + \beta < m$, $Q$ interpolates $(p_j, r_j)$ with a multiplicity of $m$. Since there are $\binom{m+1}{2}$ nonnegative integer pairs $(\alpha, \beta)$ satisfying $\alpha + \beta < m$, interpolating point $(p_j, r_j)$ with a multiplicity of $m$ implies $\binom{m+1}{2}$ constraints.

From [14], [19]

$$\phi_a(x, y) = \sum_{\alpha \in \mathbb{N}} \gamma_{a,p_j,\alpha} \psi_{p_j,\alpha}(x, y), \tag{8}$$

where $\gamma_{a,p_j,\alpha} \in \mathbb{F}_q$ are the corresponding coefficients, and

$$z^b = (z - r_j + r_j)^b = \sum_{\beta \leq b} \binom{b}{\beta} r_j^{b-\beta} (z - r_j)^{\beta}, \tag{9}$$

$Q_{\alpha,\beta}^{(p_j,r_j)}$ can be derived by substituting the above two equations into $Q = \sum_{a,b \in \mathbb{N}} Q_{ab} \phi_a z^b$ and

$$Q_{\alpha,\beta}^{(p_j,r_j)} = \sum_{a,b \geq \beta} Q_{ab} \binom{b}{\beta} \gamma_{a,p_j,\alpha} r_j^{b-\beta}. \tag{10}$$

It can be seen that to determine polynomial $Q$'s interpolation condition, the corresponding coefficients $\gamma_{a,p_j,\alpha}$ are needed. To facilitate the interpolation, they need to be pre-calculated. The following lemma shows an exception when $m = 1$.

*Lemma 2:* If the interpolation multiplicity $m = 1$, we have

$$Q_{0,0}^{(p_j,r_j)} = \sum_{a,b \in \mathbb{N}} Q_{ab} \phi_a(x_j, y_j) r_j^b. \tag{11}$$

*Proof:* When $m = 1$, $\alpha = \beta = 0$. Based on (2), we know $\psi_{p_j,0} = 1$. Further, based on (8), we have $\phi_a(x, y) = \gamma_{a,p_j,0} + \gamma_{a,p_j,1}\psi_{p_j,1} + \gamma_{a,p_j,2}\psi_{p_j,2} + \cdots$. Since $\psi_{p_j,\alpha}(x_j, y_j) = 0$ for all $\alpha$, then $\gamma_{a,p_j,0} = \phi_a(x_j, y_j)$. ■

Lemma 2 implies that when $m = 1$, pre-calculation of the corresponding coefficients can be removed and $Q$'s interpolation condition at $(p_j, r_j)$ can be simplified into its evaluation at that point. The proposed algorithms utilize this advantage to facilitate the decoding.

## III. The LCC Algorithm

With the above preliminaries, we begin introducing the LCC algorithm for Hermitian codes. The decoding begins by formulating the interpolation test-vectors.

### A. Test-Vector Formulation

In this paper, it is assumed that a Hermitian codeword is transmitted using binary phase shift keying (BPSK) modulation over a memoryless channel, e.g., the additive white Gaussian noise (AWGN) channel.

Given a received vector $\underline{\mathcal{R}} = (\mathcal{R}_0, \mathcal{R}_1, \ldots, \mathcal{R}_{n-1}) \in \mathbb{R}^n$, the reliability matrix $\Pi(\pi_{ij})_{q\times n} \in \mathbb{R}^{q\times n}$ can be obtained. Its entries are the *a posteriori* probabilities (APPs) [2] defined as

$$\pi_{ij} = \Pr[c_j = \sigma_i \mid \mathcal{R}_j], \quad i = 0, 1, \ldots, q-1, \\ j = 0, 1, \ldots, n-1. \tag{12}$$

Let $i_j^{\mathrm{I}} = \arg\max\{\pi_{ij}, \ \forall i\}$, where function $\arg\max$ returns the desirable index $i$. Further let $i_j^{\mathrm{II}} = \arg\max\{\pi_{ij}, \ \forall i$ and $i \neq i_j^{\mathrm{I}}\}$. Consequently, the most likely and the second most likely hard-decisions for $c_j$ are $r_j^{\mathrm{I}} = \sigma_{i_j^{\mathrm{I}}}$ and $r_j^{\mathrm{II}} = \sigma_{i_j^{\mathrm{II}}}$, respectively. In order to assess the reliability of each symbol's decision, we define [11]

$$\gamma_j = \frac{\pi_{i_j^{\mathrm{II}} j}}{\pi_{i_j^{\mathrm{I}} j}}, \tag{13}$$

where $\gamma_j \in (0, 1)$. When $\gamma_j \to 0$, the decision of $c_j$ is more reliable, and vice versa. By sorting all $\gamma_j$ values in ascending order, we obtain a refreshed symbol index sequence $j_0, j_1, \ldots, j_{n-1}$, where $\gamma_{j_0} < \gamma_{j_1} < \cdots < \gamma_{j_{n-1}}$. Choosing $\eta$ $(\eta < n)$ unreliable symbols, which can be realized as either $r_j^{\mathrm{I}}$ or $r_j^{\mathrm{II}}$, the indices of those symbols are contained in the set

$$\Theta = \{j_{n-\eta}, j_{n-\eta+1}, \ldots, j_{n-1}\}. \tag{14}$$

Its complementary set

$$\Theta^{\mathrm{c}} = \{j_0, j_1, \ldots, j_{n-\eta-1}\} \tag{15}$$

collects the reliable symbol indices. Consequently, the interpolation test-vectors can be formulated as

$$\underline{r}_u = (r_{j_0}^{(u)}, r_{j_1}^{(u)}, \ldots, r_{j_{n-\eta-1}}^{(u)}, r_{j_{n-\eta}}^{(u)}, \ldots, r_{j_{n-1}}^{(u)}), \tag{16}$$

where

$$r_j^{(u)} = \begin{cases} r_j^{\mathrm{I}}, & \text{if } j \in \Theta^{\mathrm{c}}, \\ r_j^{\mathrm{I}} \text{ or } r_j^{\mathrm{II}}, & \text{if } j \in \Theta. \end{cases} \tag{17}$$

Since there are two decisions for each of the $\eta$ unreliable symbols, $2^{\eta}$ interpolation test-vectors will be formulated and $u = 1, 2, \ldots, 2^{\eta}$. This test-vector formulation underpins the complexity reduction that is realized by the LCC algorithms.

### B. Common Element Interpolation

Since all test-vectors share $n - \eta$ common symbols, interpolation for points $(p_{j_0}, r_{j_0}^{\mathrm{I}})$, $(p_{j_1}, r_{j_1}^{\mathrm{I}})$, …, $(p_{j_{n-\eta-1}}, r_{j_{n-\eta-1}}^{\mathrm{I}})$ can be performed once and its outcome will be utilized by the following interpolation for each test-vector.

At the beginning, a set of polynomials are initialized by

$$\begin{aligned} \mathbf{G} &= \{Q_{\lambda+w\delta} = y^{\lambda} z^{\delta}, 0 \leq \lambda < w, \delta = 0 \text{ or } 1\} \\ &= \{1, y, \ldots, y^{w-1}, z, yz, \ldots, y^{w-1}z\}. \end{aligned} \tag{18}$$

[2] It is assumed that $\Pr[c_j = \sigma_i]$ are equal for all $i$.

Note that $|\mathbf{G}| = 2w$. For each point $(p_j, r_j^{\mathrm{I}})$ and $j \in \Theta^{\mathrm{c}}$, all polynomials' interpolation conditions are tested. In particular, given a polynomial $Q_t \in \mathbf{G}$, it can be denoted as

$$Q_t(x, y, z) = \tilde{Q}_{t,0}(x, y) + z \cdot \tilde{Q}_{t,1}(x, y). \tag{19}$$

Based on Lemma 2, its interpolation condition can be tested by $\Delta_t = Q_t(p_j, r_j^{\mathrm{I}})$ and

$$\Delta_t = \tilde{Q}_{t,0}(x_j, y_j) + r_j^{\mathrm{I}} \cdot \tilde{Q}_{t,1}(x_j, y_j). \tag{20}$$

Those polynomials with $\Delta_t = 0$ interpolate the point and do not need to be modified. The others with $\Delta_t \neq 0$ do not hold the interpolation constraint and modification is needed. For these polynomials, we first identify the minimal candidate as

$$t' = \arg\min\{Q_t \mid \Delta_t \neq 0\}. \tag{21}$$

Afterwards, the polynomials with $\Delta_t \neq 0$ but $t \neq t'$ will be modified by

$$Q'_t = \Delta_t Q_{t'} - \Delta_{t'} Q_t, \tag{22}$$

and polynomial $Q_{t'}$ will then be modified by

$$Q'_{t'} = (x - x_j) Q_{t'}. \tag{23}$$

After the modifications, all polynomials of the set interpolate point $(p_j, r_j^{\mathrm{I}})$ since $Q'_t(p_j, r_j^{\mathrm{I}}) = 0, \ \forall t$. Iterating the above process for all the common points, we obtain

$$\begin{aligned}\mathbf{G} = \{Q_t \mid Q_t(p_j, r_j^{\mathrm{I}}) = 0,\\ \forall j \in \Theta^{\mathrm{c}} \ \text{ and } t = 0, 1, \ldots, 2w - 1\}.\end{aligned} \tag{24}$$

The result will be utilized by the following uncommon element interpolation.

### *C. Uncommon Element Interpolation*

Uncommon element interpolation completes the construction of the interpolated polynomial for each test-vector. Because binary decisions were made for each unreliable symbol, the uncommon element interpolation can be performed in a *binary tree* expansion manner, as shown in Fig. 1. It starts with the outcome of the common element interpolation, i.e., $\mathbf{G}_0^{(1)}$ inherits the polynomial set $\mathbf{G}$ of (24). Including $\mathbf{G}_0^{(1)}$, there are $\eta + 1$ layers in the binary tree. We use $\mathbf{G}_s^{(s')}$ to denote the polynomial sets at layer $s$ and $s = 0, 1, \ldots, \eta$, while $s'$ identifies a particular polynomial set at the layer and $s' = 1, 2, \ldots, 2^s$. In general, polynomials of $\mathbf{G}_s^{(s')}$ will interpolate points $(p_{j_{n-\eta+s}}, r^{\mathrm{I}}_{j_{n-\eta+s}})$ and $(p_{j_{n-\eta+s}}, r^{\mathrm{II}}_{j_{n-\eta+s}})$, resulting in $\mathbf{G}_{s+1}^{(2s'-1)}$ and $\mathbf{G}_{s+1}^{(2s')}$, respectively. Based on (19), we know that for each polynomial $Q_t$ of $\mathbf{G}_s^{(s')}$, its evaluation at points $(p_{j_{n-\eta+s}}, r^{\mathrm{I}}_{j_{n-\eta+s}})$ and $(p_{j_{n-\eta+s}}, r^{\mathrm{II}}_{j_{n-\eta+s}})$ can be written as

$$\Delta_t = \tilde{Q}_{t,0}(x_{j_{n-\eta+s}}, y_{j_{n-\eta+s}}) + r^{\mathrm{I}}_{j_{n-\eta+s}} \cdot \tilde{Q}_{t,1}(x_{j_{n-\eta+s}}, y_{j_{n-\eta+s}}) \tag{25}$$

and

$$\Delta_t = \tilde{Q}_{t,0}(x_{j_{n-\eta+s}}, y_{j_{n-\eta+s}}) + r^{\mathrm{II}}_{j_{n-\eta+s}} \cdot \tilde{Q}_{t,1}(x_{j_{n-\eta+s}}, y_{j_{n-\eta+s}}), \tag{26}$$

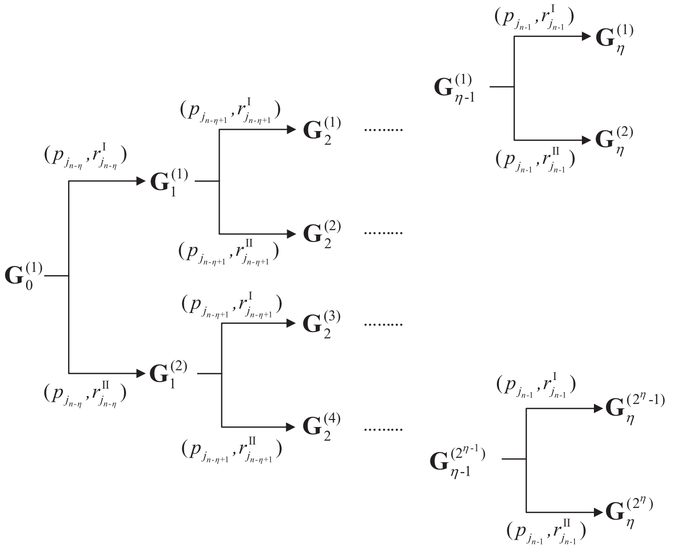
$\mathbf{G}_0^{(1)}$
$(p_{j_{n-\eta}}, r^{\mathrm{I}}_{j_{n-\eta}})$
$(p_{j_{n-\eta}}, r^{\mathrm{II}}_{j_{n-\eta}})$
$\mathbf{G}_1^{(1)}$
$\mathbf{G}_1^{(2)}$
$(p_{j_{n-\eta+1}}, r^{\mathrm{I}}_{j_{n-\eta+1}})$
$(p_{j_{n-\eta+1}}, r^{\mathrm{II}}_{j_{n-\eta+1}})$
$\mathbf{G}_2^{(1)}$ .........
$\mathbf{G}_2^{(2)}$ .........
$\mathbf{G}_2^{(3)}$ .........
$\mathbf{G}_2^{(4)}$ .........
$\mathbf{G}_{\eta\text{-}1}^{(1)}$
$(p_{j_{n-1}}, r^{\mathrm{I}}_{j_{n-1}})$
$(p_{j_{n-1}}, r^{\mathrm{II}}_{j_{n-1}})$
$\mathbf{G}_\eta^{(1)}$
$\mathbf{G}_\eta^{(2)}$
$\mathbf{G}_{\eta\text{-}1}^{(2^{\eta\text{-}1})}$
$\mathbf{G}_\eta^{(2^\eta\text{-}1)}$
$\mathbf{G}_\eta^{(2^\eta)}$

Fig. 1. Uncommon element interpolation.

respectively. Hence, $\tilde{Q}_{t,0}(x_{j_{n-\eta+s}}, y_{j_{n-\eta+s}})$ and $\tilde{Q}_{t,1}(x_{j_{n-\eta+s}}, y_{j_{n-\eta+s}})$ can be computed once and utilized by the evaluations of (25) and (26). The remaining polynomial update will be identical to that described by (21)-(23), yielding polynomial sets $\mathbf{G}_{s+1}^{(s')}$.

When the binary tree of Fig. 1 has grown, there are $2^\eta$ polynomial sets $\mathbf{G}_\eta^{(s')}$ at layer $\eta$. They correspond to the $2^\eta$ test-vectors defined by (14)-(17). The minimal polynomial is chosen from each of the sets as

$$Q^{(s')}(x, y, z) = \min\{Q_t \mid Q_t \in \mathbf{G}_\eta^{(s')}\}. \tag{27}$$

$Q^{(s')}$ is the interpolation outcome for test-vector $\underline{r}_{s'}$ and $s' = 1, 2, \ldots, 2^\eta$. They will then be factorized to find the message candidates $\hat{f}(x, y)$. Among all the decoded candidates, the one whose re-encoding produces the most likely codeword will be selected as the decoding output.

Algorithm 1 summarizes the above mentioned LCC decoding for Hermitian codes.

The above description shows that the LCC algorithm eliminates the redundant decoding computation among all test-vectors. With a multiplicity of one, the number of interpolation constraints that the decoding satisfies is equivalent to the number of distinct interpolation points implied by all test-vectors, i.e.,

$$C = (n - \eta) + \sum_{s=1}^{\eta} 2^s = n - \eta + 2^{\eta+1} - 2. \tag{28}$$

It increases exponentially with the number of unreliable symbols $\eta$, so does the decoding complexity.

## IV. The Adaptive LCC Algorithm

In order to find the intended message, the above mentioned LCC algorithm needs to decode $2^\eta$ test-vectors. However, if the channel condition improves, received information will become more reliable. Maintaining $2^\eta$ decoding test-vectors will be unnecessary since the intended message can often be

**Algorithm 1** The LCC Algorithm for Hermitian Codes

```
Input: The matrix $\Pi$ and a positive integer $\eta$;
Output: The message candidate $\hat{f}(x, y)$;
1: Determine metrics $\gamma_j$ as in (13) and define sets $\Theta$ and $\Theta^{c}$ as in (14) and (15);
2: Initialize polynomial set $\mathbf{G}$ as in (18);
3: For points $(p_j, r_j^{\mathrm{I}})$ with $j \in \Theta^{c}$ do
4:     Test the interpolation condition for each polynomial of $\mathbf{G}$ as in (20);
5:     Update the polynomials as in (21)-(23);
6: End for
7: Let $\mathbf{G}_0^{(1)} = \mathbf{G}$ and $s = 0$;
8: While $s < \eta$ do
9:     Test the interpolation condition for each polynomial of $\mathbf{G}_s^{(s')}$ as in (25)-(26);
10:    Update the polynomials as in (21)-(23);
11:    Update $s = s + 1$;
12: End while
13: Find polynomial $Q^{(s')}$ as in (27);
14: Factorize all $2^{\eta}$ polynomials $Q^{(s')}$ and select the message candidate $\hat{f}(x, y)$.
```

retrieved by decoding fewer test-vectors. Therefore, we have designed an adaptive variant of the LCC algorithm, in which the number of unreliable symbols is determined by the reliability of the received information. Instead of maintaining $\eta$ unreliable symbols, the metrics $\gamma_j$ of (13) will be used to assess each symbol's reliability in determining whether binary decisions on the symbol should be made. Consequently, the number of test-vectors will be determined by the received information's reliability. This enables the decoding complexity to be significantly reduced when the channel condition improves, e.g., the SNR increases.

Let $\Omega$ denote the symbol wise reliability threshold and $\Omega \in (0, 1)$. Recalling $\gamma_j$ of (13), if $\gamma_j < \Omega$, we consider the decision on $c_j$ to be reliable and the symbol will be realized by the most likely decision. If $\gamma_j \geq \Omega$, we consider the decision on $c_j$ to be unreliable and binary decisions will be made on the symbol. We will show that by carefully choosing $\Omega$, the adaptive LCC algorithm is able to maintain the decoding performance of the LCC algorithm. For practical concerns, we also define $\Gamma$ as the maximal number of unreliable symbols where $\Gamma \leq n$. Therefore, the number of unreliable symbols $\eta \in [1, \Gamma]$ and the number of test-vectors varies between 2 and $2^{\Gamma}$. Note that having at least one unreliable symbol is necessary to maintain the Chase decoding gain.

Hence, after sorting the symbol indices based on $\gamma_j$, $\Theta_{\Omega,\Gamma}$ is defined as the index set of the unreliable symbols, i.e.,

$$\Theta_{\Omega,\Gamma} = \{j_{n-1}, j_\chi \mid \gamma_{j_\chi} > \Omega \quad \text{and } n - \Gamma \leq \chi < n - 1\}. \tag{29}$$

There are $\eta = |\Theta_{\Omega,\Gamma}|$ unreliable symbols. The complementary set

$$\Theta_{\Omega,\Gamma}^{c} = \{0, 1, \ldots, n-1\} \setminus \Theta_{\Omega,\Gamma} \tag{30}$$

collects the indices of the reliable symbols. Therefore, in the adaptive LCC algorithm, the test-vectors can be reformulated as

$$\underline{r}^{(u)} = (r_{j_\chi}^{(u)};\ \chi = 0, 1, \ldots, n-1), \tag{31}$$

where

$$r_j^{(u)} = \begin{cases} r_j^{\mathrm{I}}, & \text{if } j \in \Theta_{\Omega,\Gamma}^{c}, \\ r_j^{\mathrm{I}} \text{ or } r_j^{\mathrm{II}}, & \text{if } j \in \Theta_{\Omega,\Gamma}. \end{cases} \tag{32}$$

There are $2^{\eta} = 2^{|\Theta_{\Omega,\Gamma}|}$ test-vectors. Furthermore, by applying the LCC algorithm of Section III, the common element interpolation will be performed for points $(p_j, r_j^{\mathrm{I}})$ where $j \in \Theta_{\Omega,\Gamma}^{c}$. Its outcome will be utilized by the uncommon element interpolation for each test-vector during which points $(p_j, r_j^{\mathrm{I}})$ and $(p_j, r_j^{\mathrm{II}})$ are interpolated where $j \in \Theta_{\Omega,\Gamma}$.

Summarizing the above description, the adaptive LCC algorithm is presented in Algorithm 2.

**Algorithm 2** The Adaptive LCC Algorithm for Hermitian Codes

```
Input: The matrix $\Pi$, the threshold $\Omega$ and the maximum number $\Gamma$;
Output: The message candidate $\hat{f}(x, y)$;
1: Determine metrics $\gamma_j$ as in (13) and define sets $\Theta_{\Omega,\Gamma}$ and $\Theta_{\Omega,\Gamma}^{c}$ as in (29) and (30);
2: Let $\eta = |\Theta_{\Omega,\Gamma}|$;
3: Perform decoding as in Steps 2 - 14 of Algorithm 1 where sets $\Theta$ and $\Theta^{c}$ are replaced by $\Theta_{\Omega,\Gamma}$ and $\Theta_{\Omega,\Gamma}^{c}$, respectively.
```

## V. Complexity Analysis

This section analyzes the complexity of the proposed LCC algorithms, where we will look into the number of finite field multiplications in a decoding event. Note that during the decoding, multiplication dominates the finite field arithmetic operations. Based on the above descriptions, we know that the LCC algorithms have three steps including the common element interpolation, the uncommon element interpolation and the factorization. We will show the complexity of each step, so that an overview of the algorithms' complexity can be reached.

### A. Common Element Interpolation Complexity

In order to analyze the common element interpolation, the following lemma is needed.

*Lemma 3:* For a polynomial $Q_t$ in the set $\mathbf{G}$ of (24), $\deg_y Q_t < n + w - \eta$.

*Proof:* The common element interpolation has $n - \eta$ iterations. Based on the polynomial update rule of (23), we know the $x$-degree of $Q_t$ can reach $n - \eta$. After the update, if the polynomial has $x$-degree greater than $w$, the transform of $x^{w+1} = y^{w} + y$ is needed to refine the polynomial in the ring $\mathbb{F}_q[x, y, z]$. Considering the initial polynomial set of (18), we know that $\deg_y Q_t \leq w - 1 + (n - \eta)\frac{w}{w+1} < n + w - \eta$. ■

With Lemma 3, the following theorem can be introduced.

*Theorem 4:* The common element interpolation has a complexity of $O((n-\eta)^3)$.

*Proof:* There are $2w$ polynomials in the set $\mathbf{G}$ participating in $n-\eta$ iterative updates. Complexity of this process is incurred by testing the interpolation condition as in (20) and updating the polynomials as in (22) or (23). Since $\deg_x \phi_a \leq w$, based on Lemma 3, we know that evaluating $\phi_a z^b$ over $(p_j, r_j)$ requires at most $n + 2w - \eta$ multiplications. For a polynomial in the set $\mathbf{G}$, it has one nonzero coefficient at the beginning and gains at most one more coefficient in each iteration. Hence, the interpolation condition test requires at most

$$2w(n+2w-\eta)\sum_{\varrho=1}^{n-\eta}\varrho \\ = w(n-\eta+1)(n-\eta)(n-\eta+2w) \approx w(n-\eta)^3$$

multiplications. The polynomial update requires at most

$$2w \cdot 2\sum_{\varrho=1}^{n-\eta}\varrho = 2w(n-\eta+1)(n-\eta) \approx 2w(n-\eta)^2$$

multiplications. ■

### B. Uncommon Element Interpolation Complexity

We now start to characterize the uncommon element interpolation complexity with the following lemma.

*Lemma 5:* For a polynomial $Q_t$ in the set $\mathbf{G}_\eta^{(s')}$, $\deg_y Q_t < n + w$.

*Proof:* The proof is similar to Lemma 3. Considering both the common and uncommon element interpolations, there are $n$ iterations for each test-vector. With the transform of $x^{w+1} = y^w + y$, we know $\deg_y Q_t \leq w - 1 + n\frac{w}{w+1} < n + w$. ■

The following theorem characterizes the uncommon element interpolation complexity.

*Theorem 6:* The uncommon element interpolation has a complexity of $O(2^{\eta+1}n^2)$.

*Proof:* We begin the analysis by looking into the complexity of updating a polynomial set $\mathbf{G}_s^{(s')}$ into $\mathbf{G}_{s+1}^{(s')}$. Assume that a polynomial of set $\mathbf{G}_s^{(s')}$ has at most $\varrho$ coefficients and $\varrho = n - \eta + 1 + s$. Based on Lemma 5, we know that testing the interpolation condition for polynomials of $\mathbf{G}_s^{(s')}$ requires at most $2w(\varrho(n+2w)+2)$ multiplications. Based on (25) and (26), we know this computation will be shared by two newly formed polynomial sets. Hence, on average each set has $w(\varrho(n+2w)+2)$ multiplications. Updating polynomials for each set requires at most $4w\varrho$ multiplications. Therefore, updating $\mathbf{G}_s^{(s')}$ into $\mathbf{G}_{s+1}^{(s')}$ requires at most $w(\varrho(n+2w+4)+2)$ multiplications.

Now, let $\varrho' = n - \eta + 1$ and $\varrho = \varrho' + s$. Fig. 1 shows there are $2^s$ polynomial sets at layer $s$. Therefore, fully expanding this binary tree requires at most $\sum_{s=1}^{\eta} 2^s w((\varrho'+s)(n+2w+4)+2)$ multiplications. Since $\sum_{s=1}^{\eta} 2^s = 2^{\eta+1} - 2$ and $\sum_{s=1}^{\eta} 2^s s = 2^{\eta+1}(\eta-1)+2$, we have

$$\sum_{s=1}^{\eta} 2^s w((\varrho'+s)(n+2w+4)+2) \\ = (n+2w+4)(2^{\eta+1}w(\varrho'+\eta-1) - 2w(\varrho'-1)) \\ +2w(2^{\eta+1}-2).$$

Replacing $\varrho'$ by $n-\eta+1$, we have

$$(n+2w+4)(2^\eta wn - 2w(n-\eta)) + 2w(2^{\eta+1}-2) \\ \approx w(2^{\eta+1}-2)(n(n+2w+4)+2),$$

where the approximation is made by assuming $n - \eta \approx n$. ■

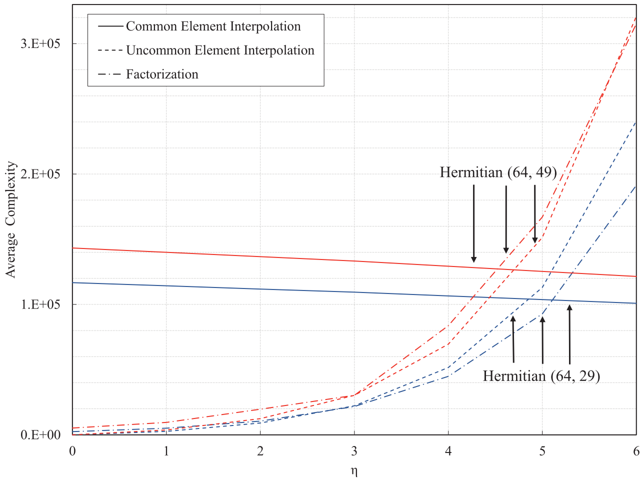

Fig. 2. LCC decoding complexity for the (64, 29) and (64, 49) Hermitian codes.

### C. Overall Complexity

Factorization is often implemented by the recursive coefficient search algorithm [16], [17] which incurs a complexity of $O(nk)$. In the proposed Chase decoding, there are at most $2^\eta$ factorizations in each decoding event. Hence, the factorization complexity will be $O(2^\eta nk)$.

The above analysis shows that when $n$ is sufficiently large and $\eta$ is small, the common element interpolation dominates the decoding complexity. However, by increasing $\eta$, the uncommon element interpolation and factorization will eventually dominate. In order to validate the above analysis, Fig. 2 shows the LCC decoding complexity for the (64, 29) and the (64, 49) Hermitian codes. We measure the number of finite field arithmetic operations that are required by the common element interpolation, the uncommon element interpolation and the factorization, respectively. It shows that the complexity of the uncommon element interpolation and factorization increases exponentially with $\eta$. For both codes, when $\eta < 5$, the common element interpolation dominates the complexity. However, when $\eta \geq 5$, the complexity is mainly caused by the uncommon element interpolation and factorization.

## VI. Decoding Performance

This section shows the decoding performance of the LCC algorithms. Our simulations are performed over the AWGN

TABLE I
DECODING COMPLEXITY FOR THE (64, 49) HERMITIAN CODE

| Sakata | GS $(m = 1)$ | KV $(l = 2)$ | KV $(l = 3)$ | LCC-1 $(\eta = 2)$ | LCC-1 $(\eta = 3)$ | LCC-1 $(\eta = 6)$ |
|---|---|---|---|---|---|---|
| $2.00 \times 10^4$ | $1.56 \times 10^5$ | $1.40 \times 10^6$ | $5.13 \times 10^6$ | $1.97 \times 10^5$ | $2.56 \times 10^5$ | $1.06 \times 10^6$ |

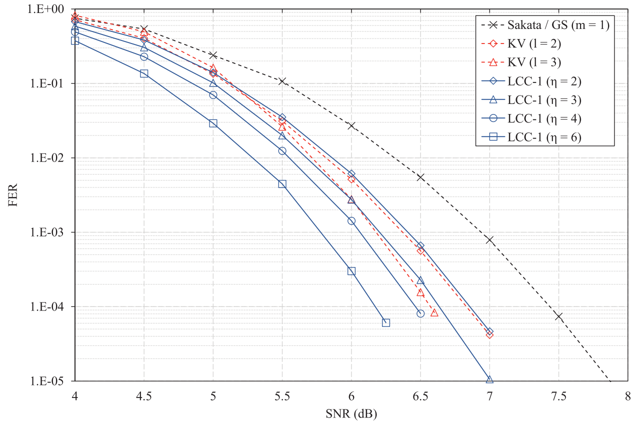

Fig. 3. LCC decoding performance of the (64, 49) Hermitian code.

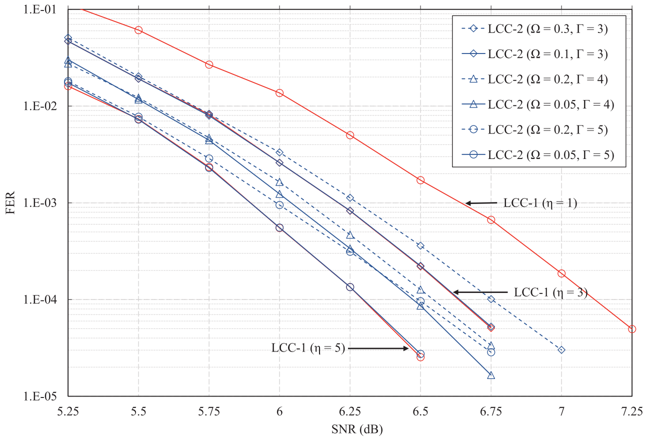

Fig. 4. Performance comparison between the LCC-1 and the LCC-2 algorithms for the (64, 49) Hermitian code.

channel using BPSK modulation. In the following discussions, we refer to the LCC algorithm (Algorithm 1) and the adaptive LCC algorithm (Algorithm 2) as LCC-1 and LCC-2, respectively.

### A. *Performance of the LCC-1 Algorithm*

Fig. 3 shows the LCC decoding performance of the (64, 49) Hermitian code that is defined in $\mathbb{F}_{16}$. It is compared with the Sakata algorithm, the GS algorithm with $m = 1$ and the KV algorithm with a maximum factorization output list size ($l$) of 2 and 3. It can be seen that the LCC-1 algorithm outperforms the Sakata and GS algorithms and its performance can be further improved by increasing the number of unreliable symbols $\eta$. The LCC-1 ($\eta = 2$) and LCC-1 ($\eta = 3$) decoding yield a similar performance as KV ($l = 2$) and KV ($l = 3$) decoding, respectively. However, Table I shows that the LCC-1 algorithm is less complex and shows the number of finite field arithmetic operations that are needed by several decoding approaches.

### B. *Comparison Between the LCC-1, LCC-2 and KV Algorithms*

Fig. 4 compares the performance of the two proposed LCC algorithms. It shows that by carefully choosing the reliability threshold $\Omega$, the LCC-2 algorithm maintains the decoding performance of the LCC-1 algorithm. For example, with $\Omega = 0.1$ and $\Gamma = 3$, the LCC-2 decoding performance matches that of the LCC-1 ($\eta = 3$). However, the LCC-2 algorithm is simpler since its number of decoding test-vectors is not always maintained at eight. Table II compares the complexity of the two LCC algorithms under different SNRs. It shows that by increasing the SNR, the complexity of the LCC-2 algorithm decreases and becomes less complex than the LCC-1 ($\eta = 3$) decoding. Fig. 4 also shows that by further increasing the maximum number of unreliable symbols, the LCC-2 algorithm outperforms the LCC-1 algorithm. For example, the LCC-2 ($\Omega = 0.2, \Gamma = 4$) decoding outperforms the LCC-1 ($\eta = 3$) decoding. Table II shows the LCC-2 ($\Omega = 0.2, \Gamma = 4$) decoding becomes less complex when the SNR $\geq 6.25$dB. With $\Gamma > \eta$, the LCC-2 algorithm can decode more test-vectors than the LCC-1 algorithm, achieving an enhanced performance. As the SNR increases, its decoding test-vectors can be less than $2^{\eta}$, leading to a smaller complexity. However, as SNR $\rightarrow \infty$, the LCC-2 algorithm will decode at most two test-vectors as $|\Theta_{\Omega,\Gamma}| = 1$. The LCC-2 algorithm's asymptotic performance will converge to that of the LCC-1 ($\eta = 1$) algorithm. Fig. 5 shows how the average number of unreliable symbols changes with the channel condition. We can see that by decreasing the reliability threshold $\Omega$, the average number converges with a higher SNR. This explains why our simulation shows that with the same $\Gamma$, a smaller $\Omega$ yields a better LCC-2 decoding performance.

Fig. 6 compares the decoding performance of the LCC-1, the LCC-2 and the KV algorithms for the (64, 39) Hermitian code. In particular, the LCC-1 algorithm is compared with the KV algorithm under the benchmark that they both satisfy the same number of interpolation constraints described by (7) and (10). Based on (28), we know LCC-1 decoding of the (64, 39) Hermitian code with $\eta = 3$ and 5 needs to satisfy 75 and 121 interpolation constraints, respectively. The number of interpolation constraints for the KV algorithm is defined by (20) in [9]. Fig. 6 shows that with the same number of interpolation constraints, the LCC-1 algorithm outperforms the KV algorithm, demonstrating performance advantage of the Chase type decoding. Note that the KV decoding performances do not differ with $C = 75$ and $C = 121$. This is because the KV decoding performance is primarily determined by its maximal

TABLE II

COMPLEXITY COMPARISON OF LCC DECODINGS FOR THE (64, 49) HERMITIAN CODE

| SNR(dB) / Algs. | 5.25 | 5.75 | 6.25 | 6.75 | 7.25 |
|---|---|---|---|---|---|
| LCC-1 $(\eta = 3)$ | $2.46 \times 10^5$ | $2.55 \times 10^5$ | $2.57 \times 10^5$ | $2.56 \times 10^5$ | $2.55 \times 10^5$ |
| LCC-2 $(\Omega = 0.1, \Gamma = 3)$ | $2.46 \times 10^5$ | $\mathbf{2.52 \times 10^5}$ | $\mathbf{2.43 \times 10^5}$ | $\mathbf{2.21 \times 10^5}$ | $\mathbf{1.96 \times 10^5}$ |
| LCC-2 $(\Omega = 0.2, \Gamma = 4)$ | $3.32 \times 10^5$ | $3.06 \times 10^5$ | $\mathbf{2.56 \times 10^5}$ | $\mathbf{2.12 \times 10^5}$ | $\mathbf{1.84 \times 10^5}$ |

TABLE III

DECODING COMPLEXITY FOR THE (64, 39) HERMITIAN CODE

| SNR(dB) / Algs. | 5.5 | 5.75 | 6 | 6.25 | 6.5 |
|---|---|---|---|---|---|
| LCC-1 $(\eta = 5)$ | $4.75 \times 10^5$ | $4.70 \times 10^5$ | $4.66 \times 10^5$ | $4.59 \times 10^5$ | $4.54 \times 10^5$ |
| LCC-2 $(\Omega = 0.4, \Gamma = 6)$ | $5.36 \times 10^5$ | $4.32 \times 10^5$ | $3.42 \times 10^5$ | $2.64 \times 10^5$ | $\mathbf{2.09 \times 10^5}$ |
| KV $(C = 121)$ | $2.21 \times 10^5$ | $2.19 \times 10^5$ | $2.17 \times 10^5$ | $2.15 \times 10^5$ | $2.12 \times 10^5$ |

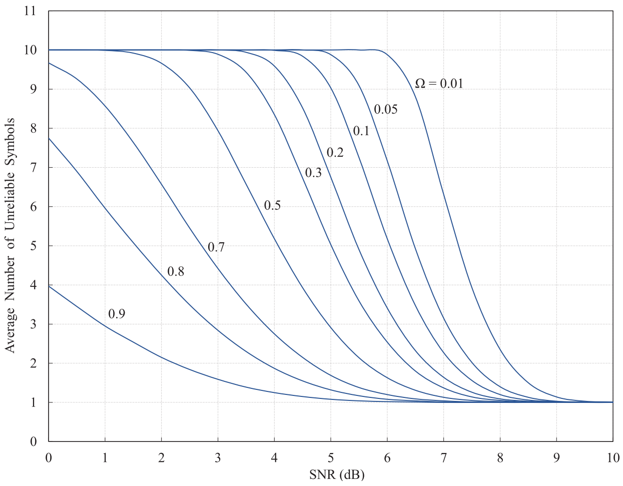

Fig. 5. Average number of unreliable symbols, $\Gamma = 10$.

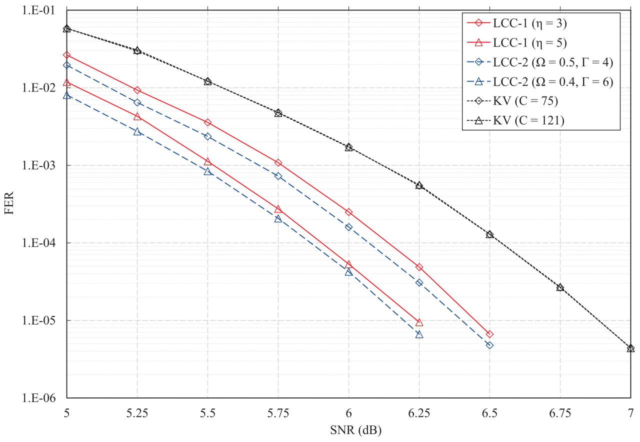

Fig. 6. Performance comparison between the LCC algorithms and the KV algorithm for the (64, 39) Hermitian code.

factorization output list size $l$. For this Hermitian code, $C = 121$ does not incur a greater $l$ than $C = 75$. Table III further compares the complexity of the three algorithms. Fig. 6 shows that with the same number of interpolation constraints the LCC decodings outperform the KV decoding. The LCC-1 algorithm is more complex than the KV algorithm, due to the fact that it has to perform $2^\eta$ factorizations while the KV algorithm performs once. The LCC-2 algorithm becomes less complex than the KV algorithm when SNR $= 6.5$dB.

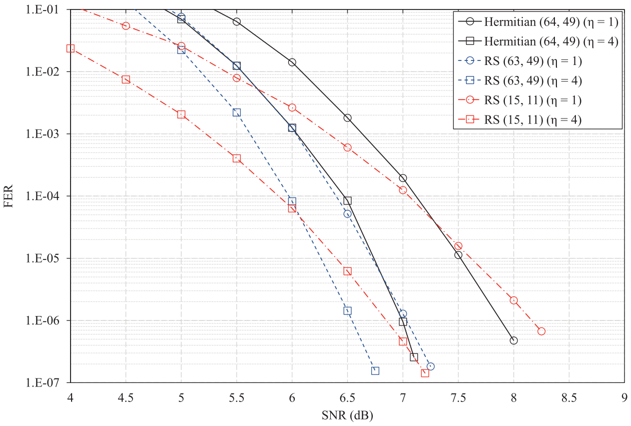

Fig. 7. LCC-1 decoding performance comparison of the (64, 49) Hermitian code and (63, 49), (15, 11) RS codes.

### *C. Comparison Between Hermitian and RS Codes*

Finally, in order to demonstrate the Hermitian code's performance advantage over RS codes, Figs. 7 and 8 compare the (64, 49) and the (512, 409) Hermitian codes with their relevant RS codes using the LCC-1 algorithm. The two Hermitian codes are defined in $\mathbb{F}_{16}$ and $\mathbb{F}_{64}$, respectively. The RS codes have similar rates as the Hermitian codes. In Fig. 7, the (15, 11) RS code is defined in the same finite field as the (64, 49) Hermitian code, while the (63, 49) RS code has a similar symbol wise codeword length as the Hermitian code. The two RS codes shown in Fig. 8 are also chosen under the same motivation when the (512, 409) Hermitian code is considered. Figs. 7 and 8 show that the (63, 49) and (511, 409) RS codes outperform the (64, 49) and (512, 409) Hermitian codes, respectively. This is because the RS codes are defined in a larger finite field with much larger bit wise codeword length. On the other hand, over the same finite field, the Hermitian codes are longer with greater error-correction capability. Hence, the two Hermitian codes outperform the (15, 11) and (63, 50) RS codes, respectively. It is important

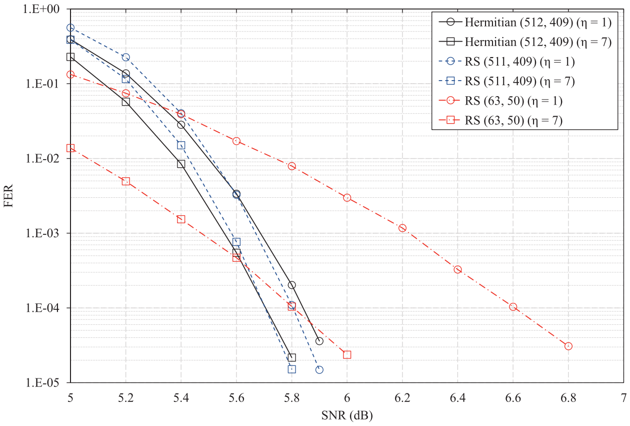

Fig. 8. LCC-1 decoding performance comparison of the (512, 409) Hermitian code and (511, 409), (63, 50) RS codes.

to point out that the Chase type decoding is search oriented. It aims to find the intended codeword through decoding trials. Intuitively, the performance gain achieved by increasing the number of decoding trials will favor codes with a smaller codebook cardinality. This conjecture is vindicated by our results. For example, in Fig. 7 the (15, 11) RS code has the smallest codebook cardinality. The LCC-1 decoding yields the largest performance gain for the code by increasing the number of unreliable symbols.

## VII. Conclusion

This paper has proposed two interpolation-based LCC decoding algorithms for Hermitian codes. By identifying $\eta$ unreliable symbols and realizing them with the two most likely decisions, the LCC algorithms will formulate $2^\eta$ interpolation test-vectors. Exploiting the similarity among the test-vectors, interpolation is performed w.r.t. the common element segment and the uncommon element segment of the test-vectors, respectively. The outcome of the common element interpolation will be shared by the uncommon element interpolation that grows the polynomial sets in a binary tree fashion. It reduces redundant computations ensuring a low-complexity feature of the decoding. With an interpolation multiplicity of one, the LCC algorithms also remove the pre-calculation of the Hermitian curve's corresponding coefficients, further facilitating the decoding. The adaptive LCC algorithm has been proposed to adjust the number of Chase decoding trials based on the reliability of the received information. Decoding complexity can be reduced with an improved channel condition. Our complexity analysis has shown that the common element interpolation complexity is $O((n-\eta)^3)$, while the uncommon element interpolation complexity is $O(2^{\eta+1}n^2)$. Therefore, by increasing $\eta$, the uncommon element interpolation will dominate the overall complexity. This has been validated by our numerical results. Simulation results on the LCC decoding algorithms have also been presented, showing the LCC algorithms can outperform the existing decoding algorithms for Hermitian codes. By carefully choosing the reliability threshold, the adaptive LCC algorithm can achieve an enhanced performance over its prototype with a smaller computational cost. LCC decoding performances for RS and Hermitian codes have also been compared, demonstrating Hermitian codes' performance advantage over RS codes.

## References

[1] V. D. Goppa, "Codes on algebraic curves," *Soviet Math. Dokl.*, vol. 24, pp. 170–172, 1981.

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

[3] S. Sakata, J. Justesen, Y. Madelung, H. E. Jensen, and T. Hoholdt, "Fast decoding of algebraic-geometric codes up to the designed minimum distance," *IEEE Trans. Inf. Theory*, vol. 41, no. 6, pp. 1672–1677, Nov. 1995.

[4] G.-L. Feng and T. R. N. Rao, "Decoding algebraic-geometric codes up to the designed minimum distance," *IEEE Trans. Inf. Theory*, vol. 39, no. 1, pp. 37–45, Jan. 1993.

[5] M. Johnston and R. A. Carrasco, "Construction and performance of algebraic-geometric codes over AWGN and fading channels," *IEE Proc.-Commun.*, vol. 152, no. 5, pp. 713–722, Oct. 2005.

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

[7] V. Guruswami and M. Sudan, "Improved decoding of Reed–Solomon and algebraic-geometry 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] L. Chen, R. Carrasco, and M. Johnston, "Soft-decision list decoding of Hermitian codes," *IEEE Trans. Commun.*, vol. 57, no. 8, pp. 2169–2176, Aug. 2009.

[10] K. Lee and M. E. O'Sullivan, "Algebraic soft-decision decoding of Hermitian codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 6, pp. 2587–2600, Jun. 2010.

[11] 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.

[12] L. Chen, "Design of an efficient list decoding system for Reed–Soloman and alebraic geometric codes," Ph.D. dissertation, Dept. Electron. Comput. Eng., Newcastle Univ., Newcastle upon Tyne, U.K., 2008.

[13] I. Blake, C. Heegard, T. Hoholdt, and V. Wei, "Algebraic-geometry codes," *IEEE Trans. Inf. Theory*, vol. 44, no. 6, pp. 2596–2618, Oct. 1998.

[14] R. Nielsen, "List decoding of linear block codes," Ph.D. dissertation, Dept. Math., Tech. Univ. Denmark, Kongens Lyngby, Denmark,2001.

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

[16] X.-W. Wu and P. H. 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.

[17] L. Chen, R. A. Carrasco, M. Johnston, and E. G. Chester, "Efficient factorisation algorithm for list decoding algebraic-geometric and Reed–Solomon codes," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Glasgow, U.K., Jun. 2007, pp. 851–856.

[18] T. Høholdt and R. Nielsen, "Decoding Hermitian codes with Sudan's algorithm," in *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes* (Lecture Notes in Computer Science), vol. 1719, M. Fossorier, S. Lin, and A. Pole, Eds. Berlin, Germany: Springer-Verlag, 1999, pp. 260–269.

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

**Siyuan Wu** received the B.Sc. degree in electronic engineering from the Guangdong University of Technology, China, in 2013, and the M.Sc. degree in information and communication engineering from Sun Yat-sen University, China, in 2017. His research interests include channel coding and wireless communications.

**Li Chen** (S'07–M'08–SM'14) received the B.Sc. degree in applied physics from Jinan University, China, in 2003, and the M.Sc. degree in communications and signal processing and the Ph.D. degree in communications engineering from Newcastle University, U.K., in 2004 and 2008, respectively. From 2007 to 2010, he was a Research Associate with Newcastle University. In 2010, he was a Lecturer with the School of Information Science and Technology, Sun Yat-sen University, China, where he became an Associate Professor in 2011. Since 2016, he has been a Professor with the School of Electronics and Communication Engineering, Sun Yat-sen University, where he is also a Deputy Dean of the School. From 2011 to 2012, he was an occasional Visiting Scholar with the Institute of Network Coding, The Chinese University of Hong Kong. In 2015, he was a Visitor at the Institute of Communication Engineering, Ulm University, Germany. From 2015 to 2016, he was a Visiting Associate Professor with the Department of Electrical Engineering, University of Notre Dame, USA. His primary research interests include information theory, channel coding, and data communications. He was a recipient of the British Overseas Research Scholarship and the Chinese Information Theory Young Researcher in 2014. He has been a Principle Investigator for three National Natural Science Foundation of China Projects and a Co-Investigator of a National Basic Research Program (973 Program) Project.

**Martin Johnston** received the B.Sc. degree (Hons.) in physics with electronics from Birmingham University, U.K., in 1999, the M.Sc. degree in electronic engineering from Staffordshire University, U.K., in 2001, and the Ph.D. degree from Newcastle University, U.K., in 2006. From 2006 to 2014, he was a Research Associate at the School of Engineering, Newcastle University, where he is currently a Lecturer. His research interests include the design of advanced error-correcting schemes and low-complexity decoding algorithms, vehicle-to-vehicle communications, and physical-layer security.