Low-Complexity Chase Decoding of Algebraic-Geometric Codes Using Koetter's Interpolation

Siyuan Wu †, Li Chen †, Martin Johnston ‡
† School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou, China, 510006
‡ School of Electrical and Electronic Engineering, Newcastle University, Newcastle-upon-Tyne, United Kingdom, NE1 7RU
Email: wusy7@mail2.sysu.edu.cn, chenli55@mail.sysu.edu.cn, martin.johnston@ncl.ac.uk

***Abstract*—Algebraic-geometric (AG) codes have long been considered as a possible candidate to replace Reed-Solomon (RS) codes. However, their decoding remains complex and infeasible to implement. Addressing this challenge, our paper proposes a low-complexity Chase (LCC) decoding algorithm for the most popular class of AG codes - Hermitian codes. The LCC decoding is realised by formulating decoding test-vectors, which allows Koetter's interpolation to be performed for common and uncommon elements. This reduces redundant computations and also removes the need to calculate the corresponding coefficients of a Hermitian curve, thus facilitating message recovery. Our simulation results show that significant coding gains can be achieved over the conventional Koetter-Vardy (KV) soft decoding algorithm, but with a much lower computational cost. Moreover, we also show that in comparison with RS codes of a similar length, Chase decoding has a more significant impact on enhancing the performance of Hermitian codes.**

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

I. Introduction

Algebraic-geometric (AG) codes were first introduced by Goppa [1] and are a class of linear block codes derived from an algebraic curve. AG codes comprise a large family including Hermitian codes, Elliptic codes and Reed-Solomon (RS) codes. The widely used RS codes can be viewed as a special class of AG codes since they are constructed from a straight line. As a result, the length of an RS code cannot exceed the size of the finite field in which it is defined, limiting its minimum distance and therefore its error-correction capability. Compared with RS codes, general AG codes have larger codeword lengths in the same finite field, leading to stronger error-correction capability.

For RS and AG codes, the conventional decoding algorithms are unique decoding algorithms, i.e. they generate a single unique decoded message. They are syndrome based and achieve codeword recovery by calculating the error locations and error magnitudes. The well-known Berlekamp-Massey (BM) algorithm [2] and the Sakata algorithm [3–5] are used to decode RS codes and Hermitian codes, respectively. However, these unique decoding algorithms can only correct errors up to half of the code's minimum distance, i.e. the number of correctable errors is $\tau \leq \lfloor (d-1)/2 \rfloor$, where d is the minimum distance of the code. To achieve a better error-correction capability for RS codes, Sudan's pioneering curve-fitting list decoding approach [6] can be deployed. However, its extra error-correcting capabiltiy only applies to RS codes of rate less than $1/3$. Guruswami and Sudan later generalised this improved decoding to both RS and AG codes of all rates and it is often named the Guruswami-Sudan (GS) algorithm [7]. Based on the GS algorithm, Koetter and Vardy presented a soft-decision list decoding algorithm for RS codes, namely the KV algorithm [8]. Soft-decision list decoding of Hermitian codes was later presented by Chen *et al.* [9] and Lee *et al.* [10], independently. However, both the GS and KV algorithms are much more complex than unique decoding algorithms. Recently, Bellorado *et al.* proposed a low-complexity Chase (LCC) decoding algorithm for RS codes [11]. They showed that a low list decoding complexity can be realised by exploiting the similarity of interpolation test-vectors. Simulation results of [11] also showed that the LCC algorithm outperforms various RS decoding algorithms, including the KV algorithm.

The LCC decoding of general AG codes appears to have not been considered in the literature. Therefore, our paper presents the first LCC decoding algorithm for Hermitian codes. With the formulation of the test-vectors, the interpolation can be performed w.r.t. the common elements and the uncommon elements, respectively. The former produces a common result that will be shared by the complete interpolation of each test-vector. The latter grows the interpolated polynomials in a *binary tree* fashion, saving the redundant computations. Unlike the KV algorithm, our proposed algorithm does not need to pre-calculate the corresponding coefficients which are essential to determining a polynomial's interpolation condition [9], saving decoding efforts. Our simulation results show that LCC decoding of Hermitian codes can outperform the KV decoding with a much lower complexity. It is also shown that in comparison with an RS code of a similar length, LCC decoding can achieve a greater coding gain by increasing the Chase decoding parameter for AG codes.

II. Preliminaries

This section presents background knowledge, including the encoding of Hermitian codes and the GS list decoding.

A. Hermitian Codes

Let $\mathbb{F}_q = \{0, 1, 2, \ldots, q-1\}$ denote the finite field of size q. Let $\mathbb{F}_q[x, y]$ and $\mathbb{F}_q[x, y, z]$ denote the bivariate and trivariate polynomial rings over $\mathbb{F}_q$, respectively. Hermitian codes are constructed from Hermitian curves. An affine Hermitian curve[1] defined over $\mathbb{F}_q$ can be written as [12]

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

where $w = \sqrt{q}$ and the curve has a genus $g = \frac{w(w-1)}{2}$. The designed minimum distance of the Hermitian code is $d = n - k - g + 1$. There are $n = w^3$ affine points $p_j = (x_j, y_j)$ that satisfy $H_w(x_j, y_j) = 0$, and a point at infinity p_∞. Let P denote the set of affine points, $P = \{p_j = (x_j, y_j), 0 \leq j \leq n-1\}$ and $|P| = w^3$.

The pole basis L_w of a Hermitian curve comprises a set of bivariate monomials $\phi_a(x, y) = x^\mu y^\nu$ with $L_w = \{\phi_a(x, y) | v_{p_\infty}(\phi_a^{-1}(x, y)) < v_{p_\infty}(\phi_{a+1}^{-1}(x, y)), a \in \mathbb{N}\}$, where $v_{p_\infty}(\phi_a^{-1}(x, y)) = w \cdot \mu + (w+1) \cdot \nu$ is the pole order of ϕ_a. For each affine point p_j, the zero basis polynomials are

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

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

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

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

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

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 according to their $(1, w_z)$-weighted degree, that is

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

where $w_z = v_{p_\infty}(\phi_{k-1}^{-1})$. 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 $\text{ord}(\phi_{a_1} z^{b_1}) < \text{ord}(\phi_{a_2} z^{b_2})$, 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 $\text{lod}(Q) = \text{ord}(\phi_{a'} z^{b'})$. Given two polynomials Q_1 and Q_2, we claim $Q_1 < Q_2$, if $\text{lod}(Q_1) < \text{lod}(Q_2)$.

The high complexity interpolation process builds an interpolated 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 starts with a polynomial set $\mathbf{G}$. Each of its polynomials grows by iteratively interpolating the n points (p_j, r_j), with a multiplicity of m. Finally, the minimal polynomial in the set will be chosen as Q for factorisation, which finds the z-roots as the decoded message candidates $\hat{f}(x, y)$ [13].

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

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

Theorem 1: Given a polynomial $Q \in \mathbb{F}_q[x, y, z]$ that has a zero of multiplicity m over the n points, if $m(n - d_{\text{H}}(\underline{c}, \underline{r})) > \deg_{1,w_z} Q$, then $Q(x, y, f) = 0$ or $(z - f) | Q(x, y, z)$ [14].

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), if polynomial Q can also 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 $\alpha + \beta < m$, then Q interpolates (p_j, r_j) with a multiplicity of m.

The relationship between a pole basis monomial ϕ_a and zero basis polynomials $\psi_{p_j,\alpha}$ can be written as [15, 16]

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

where $\gamma_{a,p_j,\alpha} \in \mathbb{F}_q$ are the corresponding coefficients. Moreover, z^b can be elaborated as

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

Using equations (8) and (9), a polynomial $Q = \sum_{a,b \in \mathbb{N}} Q_{ab} \phi_a z^b$ can be written as

$$\begin{aligned} &Q(x, y, z) \\ &= \sum_{a,b \in \mathbb{N}} Q_{ab} \left(\sum_{\alpha \in \mathbb{N}} \gamma_{a,p_j,\alpha} \psi_{p_j,\alpha} \right) \left(\sum_{\beta \leq b} \binom{b}{\beta} r_j^{b-\beta} (z - r_j)^\beta \right) \\ &= \sum_{\alpha,\beta \in \mathbb{N}} \left(\sum_{a,b \geq \beta} Q_{ab} \binom{b}{\beta} \gamma_{a,p_j,\alpha} r_j^{b-\beta} \right) \psi_{p_j,\alpha} (z - r_j)^\beta. \end{aligned} \tag{10}$$

As a result, coefficients $Q_{\alpha,\beta}^{(p_j,r_j)}$ of (7) can be written as

$$Q_{\alpha,\beta}^{(p_j,r_j)} = \sum_{a,b \geq \beta} Q_{ab} \binom{b}{\beta} \gamma_{a,p_j,\alpha} r_j^{b-\beta}. \tag{11}$$

Therefore, to determine polynomial Q's interpolation condition, the corresponding coefficients $\gamma_{a,p_j,\alpha}$ are essential. To facilitate the interpolation, they need to be pre-calculated [9]. The following lemma shows an exception when $m = 1$.

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

$$Q_{0,0}^{(p_j,r_j)} = \sum_{a,b} Q_{ab} \phi_a(x_j, y_j) r_j^b. \tag{12}$$

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

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

all α, then $\gamma_{a,p_j,0} = \phi_a(x_j, y_j)$. This completes the proof. ■

Lemma 2 implies us that with a multiplicity of one, the need to pre-calculate the corresponding coefficients is removed and Q's interpolation condition at (p_j, r_j) can be simply determined by polynomial evaluation. The proposed algorithm makes use of this advantage.

III. Low-Complexity Chase Decoding

LCC decoding starts by formulating a set of interpolation test-vectors. With this formulation, common element interpolation will be performed once and its outcome will be shared by the following uncommon element interpolation. Utilising the similarity among test-vectors, a reduction in interpolation complexity can be achieved.

A. Test-vector Formulation

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

Given a received symbol vector $\underline{R} = (R_0, R_1, \ldots, R_{n-1}) \in \mathbb{R}^n$, the reliability matrix $\mathbf{\Pi} \in \mathbb{R}^{q \times n}$ can be obtained. By assuming its rows are indexed by elements of $\mathbb{F}_q$, its entries are defined as

$$\pi_{ij} = \Pr[c_j = i | R_j], i = 0, 1, \ldots, q-1, j = 0, 1, \ldots, n-1. \tag{13}$$

Let $i_j^{\mathrm{I}} = \arg\max_{i \in \mathbb{F}_q} \{\pi_{ij}\}$ and $i_j^{\mathrm{II}} = \arg\max_{i \in \mathbb{F}_q, i \neq i_j^{\mathrm{I}}} \{\pi_{ij}\}$ denote the row indices of the largest and the second largest entries of column j, respectively. We can denote the most likely and second most likely (ML) hard-decisions for c_j as $r_j^{\mathrm{I}} = i|_{\pi_{ij} = \pi_j^{\mathrm{I}}}$ and $r_j^{\mathrm{II}} = i|_{\pi_{ij} = \pi_j^{\mathrm{II}}}$, respectively. In order to assess the reliability of each symbol's decision, we define [11]

$$\gamma_j = \frac{\pi_j^{\mathrm{II}}}{\pi_j^{\mathrm{I}}}, \tag{14}$$

where $\gamma_j \in (0, 1)$. When γ_j approaches zero, it indicates that the decision on c_j is more reliable, and vice versa. By sorting all the γ_j values in an ascending order, we obtain a refreshed symbol index sequence $j_0, j_1, \ldots, j_{n-1}$, indicating $\gamma_{j_0} < \gamma_{j_1} < \cdots < \gamma_{j_{n-1}}$.

Choosing η ($\eta < n$) unreliable symbols, the $n - \eta$ reliable symbols are identified as

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

Its complementary set Θ^{c} is

$$\Theta^{\mathrm{c}} = \{j_{n-\eta}, j_{n-\eta+1}, \ldots, j_{n-1}\}. \tag{16}$$

With the above definitions, the interpolation test-vectors can be constructed as

$$\underline{r}_u = (r_{j_0}^{(u)}, r_{j_1}^{(u)}, \ldots, r_{j_{n-\eta-1}}^{(u)}, r_{j_{n-\eta}}^{(u)}, \ldots, r_{j_{n-1}}^{(u)}) \tag{17}$$

and

$$r_j^{(u)} = \begin{cases} r_j^{\mathrm{I}}, & \text{if } j \in \Theta, \\ r_j^{\mathrm{I}} \text{ or } r_j^{\mathrm{II}}, & \text{if } j \in \Theta^{\mathrm{c}}, \end{cases} \tag{18}$$

where $u = 1, 2, \ldots, 2^\eta$ identifies a particular test-vector. This test-vector formulation underpins the complexity reduction of the LCC algorithm.

B. Common Element Interpolation

Since all the test-vectors share $n - \eta$ common symbols, interpolation for points $(p_{j_0}, r_{j_0}), (p_{j_1}, r_{j_1}), \ldots, (p_{j_{n-\eta-1}}, r_{j_{n-\eta-1}})$ can be performed once and shared.

At the beginning, a set of polynomials are initialised by

$$\begin{aligned} \mathbf{G} &= \{Q_{\lambda + w\delta} = y^\lambda z^\delta, 0 \leq \lambda < w, \delta = 0 \text{ and } 1\} \\ &= \{1, y, \ldots, y^{w-1}, z, yz, \ldots, y^{w-1}z\}. \end{aligned} \tag{19}$$

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

$$Q_t(x, y, z) = \tilde{Q}_{t,0}(x, y) + z \cdot \tilde{Q}_{t,1}(x, y). \tag{20}$$

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

$$\Delta_t = \tilde{Q}_{t,0}(x_j, y_j) + r_j \cdot \tilde{Q}_{t,1}(x_j, y_j). \tag{21}$$

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 property and modification will be needed. For those polynomials with $\Delta_t \neq 0$, we identify the minimal polymomial as

$$t' = \arg\min\{Q_t | \Delta_t \neq 0\}. \tag{22}$$

Afterwards, for the polynomials with $\Delta_t \neq 0$ but $t \neq t'$, they are modified by

$$Q'_t = \Delta_t Q_{t'} - \Delta_{t'} Q_t. \tag{23}$$

If $t = t'$, it will be modified by

$$Q'_{t'} = (x - x_j) Q_{t'}. \tag{24}$$

After the modifications, all polynomials of the set satisfy the interpolation condition for point (p_j, r_j). After interpolating all the points defined by Θ, we obtain

$$\mathbf{G} = \{Q_t | Q_t(p_j, r_j) = 0, \forall j \in \Theta \text{ and } t = 0, 1, \ldots, 2w-1\}. \tag{25}$$

This will be utilised by the following uncommon element interpolation.

C. Uncommon Element Interpolation

Uncommon element interpolation completes the building of the interpolated polynomial Q for each test-vector. Because of the binary decision on each unreliable symbol, it can also grow the polynomial sets in a *binary tree* fashion. The uncommon element interpolation is shown as in Fig.1. It starts with the outcome of the common element interpolation, i.e., $\mathbf{G}_0^{(1)}$ inherits $\mathbf{G}$ of (25). 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 ($s = 0, 1, \ldots, \eta$). s' identifies a particular polynomial set at that layer and $s' = 1, 2, \ldots, 2^s$. In general, based on $\mathbf{G}_s^{(s')}$, one can interpolate points $(p_{j_{n-\eta+s}}, r_{j_{n-\eta+s}}^{\mathrm{I}})$ and

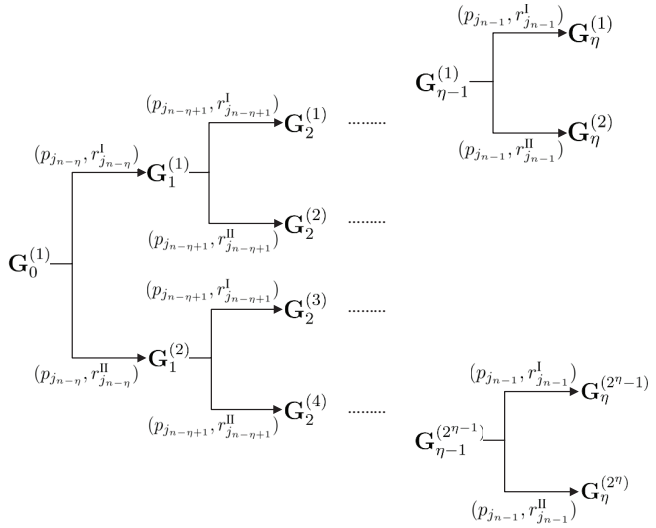

Fig. 1. Uncommon element interpolation.

$(p_{j_{n-\eta+s}}, r^{\mathrm{II}}_{j_{n-\eta+s}})$, resulting in polynomial sets $\mathbf{G}^{(2s'-1)}_{s+1}$ and $\mathbf{G}^{(2s')}_{s+1}$, respectively. By observing the polynomial structure of (20), this process can be facilitated. 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

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

and

$$\Delta_t = \tilde{Q}_{t,0}(x_{j_{n-\eta+s}}, y_{j_{n-\eta+s}}) + r^{\mathrm{II}}_{j_{n-\eta+s}} \cdot \tilde{Q}_{t,1}(x_{j_{n-\eta+s}}, y_{j_{n-\eta+s}}), \quad (27)$$

respectively. Therefore, $\tilde{Q}_{t,0}(x_{j_{n-\eta+s}}, y_{j_{n-\eta+s}})$ and $\tilde{Q}_{t,1}(x_{j_{n-\eta+s}}, y_{j_{n-\eta+s}})$ can be computed once and utilised by the evaluations of (26) and (27). The remaining polynomial update will be identical to that described by (22)-(24), yielding polynomial sets $\mathbf{G}^{(s')}_{s+1}$.

After layer η has materialised, 2^{η} polynomial sets $\mathbf{G}^{(s')}_{\eta}$ are obtained. They correspond to the 2^{η} test-vectors defined by (15)-(18). The minimal polynomial is chosen from each of the sets as

$$Q(x, y, z) = \min\{Q_t | Q_t \in \mathbf{G}^{(s')}_{\eta}\}. \quad (28)$$

They will then be factorised to retrieve the message candidates. Among all the possible 2^{η} candidates, the one that corresponds to the ML codeword will be selected as the decoding output.

Summarising the above description, LCC decoding of Hermitian codes is presented as in Algorithm 1.

IV. Performance and Complexity Analyses

This section presents the performance and complexity analyses for the LCC algorithm. The frame error rate (FER) performance is obtained over the AWGN channel using BPSK modulation. The complexity is measured as the average number of arithmetic finite field operations required to decode a codeword frame.

Fig. 2 shows the LCC decoding performance for the (64, 49) Hermitian code. The Sakata, GS ($m = 1$) and KV decoding

Algorithm 1 The LCC algorithm for Hermitian codes

Input: The reliability matrix $\mathbf{\Pi}$ and a positive integer η;
Output: The message candidate $\hat{f}(x, y)$;

1: Determine metrics γ_j as in (14) and define Θ and Θ^{c} as in (15) and (16);
2: Initialise polynomial set $\mathbf{G}$ as in (19);
3: **For** points (p_j, r_j) with $j \in \Theta$ **do**
4: Testify the interpolation condition for each polynomial of $\mathbf{G}$ as in (21);
5: Update the polynomials as in (22)-(24);
6: **End for**
7: Let $\mathbf{G}^{(1)}_{0} = \mathbf{G}$ and $s = 0$;
8: **While** $s < \eta$ **do**
9: Testify the interpolation condition for each polynomial of $\mathbf{G}^{(s')}_{s}$ as in (26)-(27);
10: Update the polynomials as in (22)-(24);
11: Update $s = s + 1$;
12: **End while**
13: Find polynomial Q of each set $\mathbf{G}^{(s')}_{\eta}$ as in (28);
14: Factorise all the 2^{η} minimal polynomials and select the message candidate $\hat{f}(x, y)$.

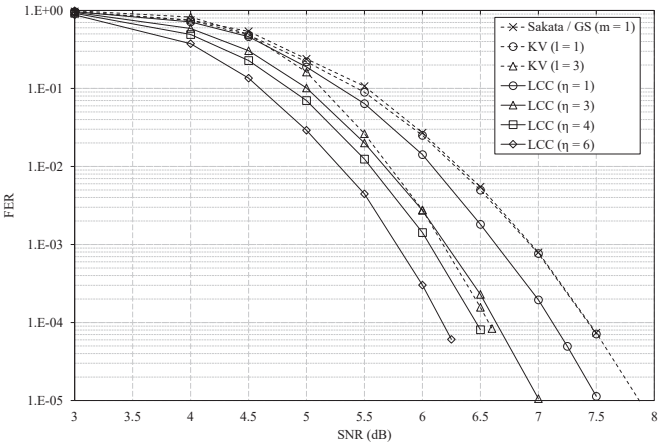

Fig. 2. Performance of the (64, 49) Hermitian code over the AWGN channel.

performances are given as comparison benchmarks. In particular, we compare the LCC algorithm and the KV algorithm under the constraint that both of the soft decodings incur a similar interpolation cost (the total number of interpolation constraints) which is a good indicator for the complexity of Koetter's interpolation. For example, parameterised by the z degree of Q, i.e., $l = \deg_z Q$, KV decoding with $l = 1$ and $l = 3$ has a similar interpolation cost as LCC decoding with $\eta = 1$ and $\eta = 3$, respectively. As shown in Fig. 2, the LCC decoding with $\eta = 1$ yields a better performance than the KV decoding with $l = 1$. But KV decoding with $l = 3$ slightly outperforms the LCC decoding with $\eta = 3$. However, it is more complex as indicated by Table I. Table I shows that the LCC decoding is far simpler than the KV decoding. For example, LCC decoding with $\eta = 3$ remains far simpler than KV decoding with $l = 3$. It is even simpler than KV decoding

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

Sakata	GS ($m = 1$)	KV ($l = 1$)	KV ($l = 3$)	LCC ($\eta = 1$)	LCC ($\eta = 3$)	LCC ($\eta = 6$)
2.00×10^4	9.47×10^4	3.01×10^5	4.54×10^6	1.04×10^5	1.63×10^5	6.46×10^5

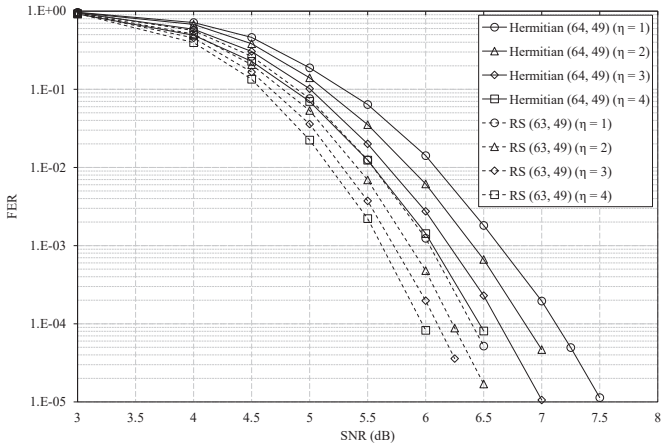

Fig. 3. Comparison of LCC decoding of RS and Hermitian codes.

with $l = 1$, but performs better. In general, the LCC decoding outperforms all the benchmarks and its performance can be enhanced by increasing η, which leads to more interpolation test-vectors. Of course, this is also at the cost of complexity as shown in Table I.

Comparing with an RS code of a similar length and rate, the Hermitian codebook has a much smaller cardinality due to the fact that it is written over a smaller finite field. A smaller codebook cardinality can favor Chase decoding which is search oriented. Fig. 3 consolidates such a conjecture by comparing the LCC decoding performance for the (63, 49) RS code and the (64, 49) Hermitian code. The RS code is defined in $\mathbb{F}_{64}$ while the Hermitian code is defined in $\mathbb{F}_{16}$. Fig. 3 shows that by increasing η, greater coding gains can be achieved for the Hermitian code. For example, by increasing η from one to four, the LCC decoding achieves a 0.7 dB coding gain for the Hermitian code. While for the RS code, it is only 0.4 dB. Note that this Hermitian code does not outperform the RS code, since it is not maximal distance separable and remains much shorter in bits.

V. CONCLUSIONS

This paper has introduced the first LCC decoding algorithm for the most popular class of AG codes, the Hermitian codes. The interpolation test-vectors are formulated by considering both the most likely and the second most likely decisions for the unreliable symbols. Such a formulation allows the interpolation to grow the interpolated polynomials in a *binary tree* fashion, reducing redundant computations. Moreover, our proposed algorithm does not require the pre-calculation of the corresponding coefficients, which is essential for determining the interpolation condition. Our simulation results have shown that significant performance gains can be achieved over the Sakata, GS and KV algorithms. It is also far less complex than the KV algorithm. Finally, we have also demonstrated that the LCC decoding of a Hermitian code can obtain a greater Chase decoding gain than an RS code of a similar length and rate.

ACKNOWLEDGEMENT

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

REFERENCES

[1] V. D. Goppa, "Codes on algebraic curves," *Soviet Math*, vol. Dol. 24, pp. 170–172, 1981.
[2] J. L. Massey, "Shift-register synthesis and BCH decoding," *IEEE Trans. Inform. 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. Inform. Theory*, vol. 41, no. 5, pp. 1672–1677, Sept. 1995.
[4] G. Feng and T. Rao, "Decoding algebraic-geometric codes up to the designed minimum distance," *IEEE Trans. Inform. Theory*, vol. 39, no. 1, pp. 37–46, Jan. 1993.
[5] M. Johnston and R. 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. Compl.*, 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. Inform. Theory*, vol. 45, no. 6, pp. 1757–1767, Sept. 1999.
[8] R. Koetter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," *IEEE Trans. Inform. Theory*, vol. 49, no. 11, pp. 2809–2825, Nov. 2003.
[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. O'Sullivan, "Algebraic soft-decision decoding of Hermitian codes," *IEEE Trans. Inform. 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. Inform. Theory*, vol. 56, no. 3, pp. 945–959, Mar. 2010.
[12] I. Blake, C. Heegard, T. Hoholdt and V. Wei, "Algebraic-geometry codes," *IEEE Trans. Inform. Theory*, vol. 44, no. 6, pp. 2596–2618, Oct. 1998.
[13] X. Wu and P. Siegel, "Efficient root-finding algorithm with application to list decoding of algebraic-geometric codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 6, pp. 2579–2587, Sept. 2001.
[14] T. Høholdt and R. 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, H. I. N. Fossorier, S. Lin, and A. Pole, Ed. Berlin, Germany: Springer-Verlag, 1999, pp. 260–269.
[15] R. R. Nielsen, *List Decoding of Linear Block Codes*. Lyngby, Denmark: Tech. Univ. Denmark, 2001.
[16] 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.