List Decoding Algorithm based on Voting in Gröbner Bases for General One-Point AG Codes

Ryutaroh Matsumoto[*], Diego Ruano[†], and Olav Geil[†]

April 22, 2013

Abstract

We generalize the unique decoding algorithm for one-point AG codes over the Miura-Kamiya C_{ab} curves proposed by Lee, Bras-Amorós, and O'Sullivan [31] to general one-point AG codes, without any assumption. We also extend their unique decoding algorithm to list decoding, modify it so that it can be used with the Feng-Rao improved code construction, prove equality between its error correcting capability and half the minimum distance lower bound by Andersen and Geil [3] that has not been done in the original proposal except for one-point Hermitian codes, remove the unnecessary computational steps so that it can run faster, and analyze its computational complexity in terms of multiplications and divisions in the finite field. As a unique decoding algorithm, the proposed one is as fast as the BMS algorithm for one-point Hermitian codes, and as a list decoding algorithm it is much faster than the algorithm by Beelen and Brander [5].

Keywords: algebraic geometry code, Gröbner basis, list decoding
MSC 2010: Primary: 94B35; Secondary: 13P10, 94B27, 14G50

1 Introduction

We consider the list decoding of one-point algebraic geometry (AG) codes. Guruswami and Sudan [24] proposed the well-known list decoding algorithm for

[*]Department of Communications and Computer Enginnering, Tokyo Instiutte of Technology, 152-8550 Japan.
[†]Department of Mathematical Sciences, Aalborg University, Denmark.

one-point AG codes, which consists of the interpolation step and the factorization step. The interpolation step has large computational complexity and many researchers have proposed faster interpolation steps, see [5, Figure 1].

By modifying the unique decoding algorithm [31] for primal one-point AG codes, we propose another list decoding algorithm based on voting in Gröbner bases whose error correcting capability is higher than [24] and whose computational complexity is smaller than [5, 24] in many cases. A decoding algorithm for primal one-point AG codes was proposed in [32], which was a straightforward adaptation of the original Feng-Rao majority voting for the dual AG codes [16] to the primal ones. The Feng-Rao majority voting in [32] for one-point primal codes was generalized to multi-point primal codes in [6, Section 2.5]. The one-point primal codes can also be decoded as multi-point dual codes with majority voting [4, 12, 13], whose faster version was proposed in [44] for the multi-point Hermitian codes. Lee, Bras-Amorós, and O'Sullivan [31] proposed another unique decoding (not list decoding) algorithm for primal codes based on the majority voting inside Gröbner bases. The module used by them [31] is a curve theoretic generalization of one used for Reed-Solomon codes in [2] that is a special case of the module used in [29]. An interesting feature in [31] is that it did not use differentials and residues on curves for its majority voting, while they were used in [6, 32]. The above studies [6, 31, 32] dealt with the primal codes. We recently proved in [23] that the error-correcting capabilities of [31, 32] are the same. The earlier papers [10, 39] suggest that central observations in [3, 23, 32] were known to the Dutch group, which is actually the case [11]. Chen [8], Elbrønd Jensen et al. [15] and Bras-Amorós and O'Sullivan [7] studied the error-correcting capability of the Feng-Rao [16] or the BMS algorithm [45, 46] with majority voting beyond half the designed distance that are applicable to the dual one-point codes.

There was room for improvements in the original result [31], namely, (a) they have not clarified the relation between its error-correcting capability and existing minimum distance lower bounds except for the one-point Hermitian codes, (b) they have not analyzed the computational complexity, (c) they assumed that the maximum pole order used for code construction is less than the code length, and (d) they have not shown how to use the method with the Feng-Rao improved code construction [17]. We shall (1) prove that the error-correcting capability of the original proposal is always equal to half of the bound in [3] for the minimum distance of one-point primal codes (Proposition 6), (2) generalize their algorithm to work with any one-point AG codes, (3) modify their algorithm to a list decoding algorithm, (4) remove the assumptions (c) and (d) above, (5) remove unnecessary computational steps from the original proposal, (6) analyze the computational complexity in terms of the number of multiplications and divisions in the finite field. The proposed algorithm is implemented on the Singular computer algebra system [9], and we verified that the proposed algorithm can correct more errors than [5, 24] with manageable computational complexity.

This paper is organized as follows: Section 2 introduces notations and relevant facts. Section 3 improves [31] in various ways, and the differences to the original [31] are summarized in Section 3.8. Section 4 shows that the proposed modification to [31] works as claimed. Section 5 compares its computational complexity with the conventional methods. Section 6 concludes the paper. Part of this paper was presented at 2012 IEEE International Symposium on Information Theory, Cambridge, MA, USA, July 2012 [22].

2 Notation and Preliminary

Our study heavily relies on the standard form of algebraic curves introduced independently by Geil and Pellikaan [20] and Miura [38], which is an enhancement of earlier results [37, 41]. Let $F/\mathbf{F}_q$ be an algebraic function field of one variable over a finite field $\mathbf{F}_q$ with q elements. Let g be the genus of F. Fix $n+1$ distinct places $Q, P_1, \ldots, P_n$ of degree one in F and a nonnegative integer u. We consider the following one-point algebraic geometry (AG) code

$$C_u = \{\mathrm{ev}(f) \mid f \in \mathcal{L}(uQ)\} \tag{1}$$

where $\mathrm{ev}(f) = (f(P_1), \ldots, f(P_n))$. Suppose that the Weierstrass semigroup $H(Q)$ at Q is generated by $a_1, \ldots, a_t$, and choose t elements $x_1, \ldots, x_t$ in F whose pole divisors are $(x_i)_\infty = a_i Q$ for $i = 1, \ldots, t$. We do *not* assume that a_1 is the smallest among $a_1, \ldots, a_t$. Without loss of generality we may assume the availability of such $x_1, \ldots, x_t$, because otherwise we cannot find a basis of C_u for every u. Then we have that $\mathcal{L}(\infty Q) = \cup_{i=1}^{\infty} \mathcal{L}(iQ)$ is equal to $\mathbf{F}_q[x_1, \ldots, x_t]$ [41]. We express $\mathcal{L}(\infty Q)$ as a residue class ring $\mathbf{F}_q[X_1, \ldots, X_t]/I$ of the polynomial ring $\mathbf{F}_q[X_1, \ldots, X_t]$, where $X_1, \ldots, X_t$ are transcendental over $\mathbf{F}_q$, and I is the kernel of the canonical homomorphism sending X_i to x_i. Geil and Pellikaan [20], Miura [38] identified the following convenient representation of $\mathcal{L}(\infty Q)$ by using Gröbner basis theory [1]. The following review is borrowed from [34]. Hereafter, we assume that the reader is familiar with the Gröbner basis theory in [1].

Let $\mathbf{N}_0$ be the set of nonnegative integers. For $(m_1, \ldots, m_t), (n_1, \ldots, n_t) \in \mathbf{N}_0^t$, we define the weighted reverse lexicographic monomial order $\succ$ such that $(m_1, \ldots, m_t) \succ (n_1, \ldots, n_t)$ if $a_1 m_1 + \cdots + a_t m_t > a_1 n_1 + \cdots + a_t n_t$, or $a_1 m_1 + \cdots + a_t m_t =$

$a_1 n_1 + \cdots + a_t n_t$, and $m_1 = n_1$, $m_2 = n_2$, $\ldots$, $m_{i-1} = n_{i-1}$, $m_i < n_i$, for some $1 \le i \le t$. Note that a Gröbner basis of I with respect to $\succ$ can be computed by [41, Theorem 15], [47], [49, Theorem 4.1] or [51, Proposition 2.17], starting from any affine defining equations of $F/\mathbf{F}_q$.

Example 1 *According to Høholdt and Pellikaan [25, Example 3.7],*

$$u^3 v + v^3 + u = 0$$

is an affine defining equation for the Klein quartic over $\mathbf{F}_8$*. There exists a unique* $\mathbf{F}_8$*-rational place* Q *such that* $(v)_\infty = 3Q$*,* $(uv)_\infty = 5Q$*, and* $(u^2 v)_\infty = 7Q$*. The numbers* 3, 5 *and* 7 *is the minimal generating set of the Weierstrass semigroup at* Q*. Choosing* v *as* x_1*,* uv *as* x_2 *and* $u^2 v$ *as* x_3*, by [49, Theorem 4.1] we can see that the standard form of the Klein quartic is given by*

$$X_2^2 + X_3 X_1, X_3 X_2 + X_1^4 + X_2, X_3^2 + X_2 X_1^3 + X_3,$$

which is the reduced Gröbner basis with respect to the monomial order $\succ$*. We can see that* $a_1 = 3$*,* $a_2 = 5$*, and* $a_3 = 7$*.*

Example 2 *Consider the function field* $\mathbf{F}_9(u_1, v_2, v_3)$ *with relations*

$$v_2^3 + v_2 = u_1^4, \quad v_3^3 + v_3 = (v_2/u_1)^4. \tag{2}$$

This is the third function field in the asymptotically good tower introduced by Garcia and Stichtenoth [19]. Substituting v_2 *with* $u_1 u_2$ *and* v_3 *with* $u_2 u_3$ *in Eq. (2) we have affine defining equations*

$$u_1^2 u_2^3 + u_2 - u_1^3 = 0, \quad u_2^2 u_3^3 + u_3 - u_2^3 = 0.$$

The function u_1 *has a unique pole* Q *in* $\mathbf{F}_9(u_1, u_2, u_3) = \mathbf{F}_9(u_1, v_2, v_3)$*. The minimal generating set of the Weiestrass semigroup* $H(Q)$ *at* Q *is* 9, 12, 22, 28, 32 *and* 35 *[52, Example 4.11]. It has genus* 22 *and* 77 $\mathbf{F}_9$*-rational points different from* Q *[19].*

Define six functions $x_1 = u_1$*,* $x_2 = u_1 u_2$*,* $x_3 = u_1^2 u_2 u_3$*,* $x_4 = u_1^3 u_2^2 u_3^2$*,* $x_5 = ((u_1 u_2)^2 + 1) u_2 u_3$ *and* $x_6 = ((u_1 u_2)^2 + 1) u_2^2 u_3^2$*. We have* $(x_1)_\infty = 9Q$*,* $(x_2)_\infty = 12Q$*,* $(x_3)_\infty = 22Q$*,* $(x_4)_\infty = 35Q$*,* $(x_5)_\infty = 28Q$ *and* $(x_6)_\infty = 32Q$ *[50]. From this information and [49, Theorem 4.1] we can compute the* 15 *polynomials in the reduced Gröbner basis of the ideal* $I \subset \mathbf{F}_9[X_1, \ldots, X_6]$ *defining* $\mathcal{L}(\infty Q)$ *as* $\{X_2^3 - X_1^4 + X_2, X_5 X_2 - X_3 X_1^2, X_6 X_2 - X_4 X_1, X_3^2 - X_4 X_1, X_3 X_2^2 - X_5 X_1^2 + X_3, X_5 X_3 - X_6 X_1^2,$

$X_6X_3 - X_1^6 + X_5X_1^2 + X_2X_1^2$, $X_5^2 - X_4X_2X_1 - X_6$, $X_4X_3 - X_2X_1^5 + X_3X_1^3 + X_2^2X_1$, $X_4X_2^2 - X_6X_1^3 + X_4$, $X_6X_5 - X_2^2X_1^4 + X_3X_2X_1^2 + X_5$, $X_5X_4 - X_1^7 + X_5X_1^3 + X_2X_1^3$, $X_6^2 - X_5X_1^4 + X_4X_2X_1 + X_3X_1^2 + X_6$, $X_6X_4 - X_3X_1^5 + X_6X_1^3 + X_3X_2X_1$, $X_4^2 - X_3X_2X_1^4 + X_4X_1^3 + X_5X_1^2 - X_3\}$. *Note that polynomials in the above Gröbner basis are in the ascending order with respect to the monomial order $\prec$ while terms in each polynomial are in the descending order with respect to $\prec$.*

For $i = 0, \ldots, a_1 - 1$, we define $b_i = \min\{m \in H(Q) \mid m \equiv i \pmod{a_1}\}$, and L_i to be the minimum element $(m_1, \ldots, m_t) \in \mathbf{N}_0^t$ with respect to $\prec$ such that $a_1m_1 + \cdots + a_tm_t = b_i$. Note that b_i's are the well-known Apéry set [40, Lemmas 2.4 and 2.6] of the numerical semigroup $H(Q)$. Then we have $\ell_1 = 0$ if we write L_i as $(\ell_1, \ldots, \ell_t)$. For each $L_i = (0, \ell_{i2}, \ldots, \ell_{it})$, define $y_i = x_2^{\ell_{i2}} \cdots x_t^{\ell_{it}} \in \mathcal{L}(\infty Q)$.

The footprint of I, denoted by $\Delta(I)$, is $\{(m_1, \ldots, m_t) \in \mathbf{N}_0^t \mid X_1^{m_1} \cdots X_t^{m_t}$ is not the leading monomial of any nonzero polynomial in I with respect to $\prec\}$, and define $\Omega_0 = \{x_1^{m_1} \cdots x_t^{m_t} \mid (m_1, \ldots, m_t) \in \Delta(I)\}$. Then Ω_0 is a basis of $\mathcal{L}(\infty Q)$ as an $\mathbf{F}_q$-linear space [1], two distinct elements in Ω_0 have different pole orders at Q, and

$$\begin{aligned} \Omega_0 &= \{x_1^m x_2^{\ell_2} \cdots, x_t^{\ell_t} \mid m \in \mathbf{N}_0, (0, \ell_2, \ldots, \ell_t) \in \{L_0, \ldots, L_{a_1-1}\}\} \\ &= \{x_1^m y_i \mid m \in \mathbf{N}_0, i = 0, \ldots, a_1 - 1\}. \end{aligned} \quad (3)$$

Equation (3) shows that $\mathcal{L}(\infty Q)$ is a free $\mathbf{F}_q[x_1]$-module with a basis $\{y_0, \ldots, y_{a_1-1}\}$. Note that the above structured shape of Ω_0 reflects the well-known property of every weighted reverse lexicographic monomial order, see the paragraph preceding to [14, Proposition 15.12].

Example 3 *For the curve in Example 1, we have $y_0 = 1$, $y_1 = x_3$, $y_2 = x_2$.*

Let v_Q be the unique valuation in F associated with the place Q. The semigroup $H(Q)$ is equal to $\{ia_1 - v_Q(y_j) \mid 0 \le i, 0 \le j < a_1\}$ [40, Lemma 2.6]. By [34, Proposition 3.18], for each nongap $s \in H(Q)$ there is a unique monomial $x_1^i y_j \in \Omega_0$ with $0 \le j < a_1$ such that $-v_Q(x_1^i y_j) = s$, and let us denote this monomial by φ_s. Let $\Gamma \subset H(Q)$, and we may consider the one-point codes

$$C_\Gamma = \langle \{\mathrm{ev}(\varphi_s) \mid s \in \Gamma\} \rangle, \quad (4)$$

where $\langle \cdot \rangle$ denotes the $\mathbf{F}_q$-linear space spanned by $\cdot$. Since considering linearly dependent rows in a generator matrix has no merit, we assume

$$\Gamma \subseteq \widehat{H}(Q), \quad (5)$$

where $\widehat{H}(Q) = \{u \in H(Q) \mid C_u \neq C_{u-1}\}$. One motivation for considering these codes is that it was shown in [3] how to increase the dimension of the one-point codes without decreasing the lower bound d_{AG} for the minimum distance. The bound $d_{\mathrm{AG}}(C_\Gamma)$ is defined for C_Γ as follows [3]: For $s \in \Gamma$, let

$$\lambda(s) = \sharp\{j \in H(Q) \mid j + s \in \widehat{H}(Q)\}. \tag{6}$$

Then $d_{\mathrm{AG}}(C_\Gamma) = \min\{\lambda(s) \mid s \in \Gamma\}$. It is proved in [21] that d_{AG} gives the same estimate for the minimum distance as the Feng-Rao bound [16] for one-point dual AG codes when both d_{AG} and the Feng-Rao bound can be applied, that is, when the dual of a one-point code is isometric to a one-point code. Furthermore, it is also proved in [21] that $d_{\mathrm{AG}}(C_\Gamma)$ can be obtained from the bounds in [4, 12, 13], hence d_{AG} can be understood as a particular case of these bounds [4, 12, 13].

3 Procedure of New List Decoding based on Voting in Gröbner Bases

3.1 Overall Structure

Suppose that we have a received word $\vec{r} \in \mathbf{F}_q^n$. We shall modify the unique decoding algorithm proposed by Lee et al. [31] so that we can find all the codewords in C_Γ in Eq. (4) within the Hamming distance τ from $\vec{r}$. The overall structure of the modified algorithm is as follows:

1. Precomputation before getting a received word $\vec{r}$,
2. Initialization after getting a received word $\vec{r}$,
3. Termination criteria of the iteration, and
4. Main part of the iteration.

Steps 2 and 4 are based on [31]. Steps 1 and 3 are not given in [31]. Each step is described in the following subsections in Section 3. We shall analyze time complexity except the precomputation part of the algorithm.

3.2 Modified Definitions for the Proposed Modification

We retain notations from Section 2. In this subsection, we modify notations and definitions in [31] to describe the proposed modification to their algorithm. We also introduce several new notations. Define a set $\Omega_1 = \{x_1^i y_j z^k \mid 0 \le i, 0 \le j < a_1, k = 0, 1\}$. Our Ω_1 is Ω in [31]. Recall also that $\Omega_0 = \{\varphi_s \mid s \in H(Q)\}$.

Since the $\mathbf{F}_q[x_1]$-module $\mathcal{L}(\infty Q)z \oplus \mathcal{L}(\infty Q)$ has a free basis $\{y_j z, y_j \mid 0 \le j < a_1\}$, we can regard Ω_1 as the set of monomials in the Gröbner basis theory for modules. We introduce a monomial order on Ω_1 as follows. For given two monomials $x_1^{i_1} y_j z^{i_{t+1}}$ and $x_1^{i'_1} y_{j'} z^{i'_{t+1}}$, first rewrite y_j and $y_{j'}$ by $x_2, \ldots, x_t$ defined in Section 2 and get $x_1^{i_1} y_j z^{i_{t+1}} = x_1^{i_1} x_2^{i_2} \cdots x_t^{i_t} z^{i_{t+1}}$ and $x_1^{i'_1} y_{j'} z^{i'_{t+1}} = x_1^{i'_1} x_2^{i'_2} \cdots x_t^{i'_t} z^{i'_{t+1}}$. For a nongap $s \in H(Q)$, we define the monomial order $x_1^{i_1} x_2^{i_2} \cdots x_t^{i_t} z^{i_{t+1}} <_s x_1^{i'_1} x_2^{i'_2} \cdots x_t^{i'_t} z^{i'_{t+1}}$ parametrized by s if $i_{t+1} s - v_Q(x_1^{i_1} x_2^{i_2} \cdots x_t^{i_t}) < i'_{t+1} s - v_Q(x_1^{i'_1} x_2^{i'_2} \cdots x_t^{i'_t})$ or $i_{t+1} s - v_Q(x_1^{i_1} x_2^{i_2} \cdots x_t^{i_t}) = i'_{t+1} s - v_Q(x_1^{i'_1} x_2^{i'_2} \cdots x_t^{i'_t})$ and $i_1 = i'_1, i_2 = i'_2, \ldots, i_{\ell-1} = i'_{\ell-1}$ and $i_\ell > i'_\ell$ for some $1 \le \ell \le t+1$. Observe that the restriction of $<_s$ to Ω_0 is equal to $\prec$ defined in Section 2. In what follows, every Gröbner basis, leading term, and leading coefficient is obtained by considering the Gröbner basis theory for modules, not for ideals.

For $f \in \mathcal{L}(\infty Q)z \oplus \mathcal{L}(\infty Q)$, $\gamma(f)$ denotes the number of nonzero terms in f when f is expressed as an $\mathbf{F}_q$-linear combination of monomials in Ω_1. $\gamma_{\neq 1}(f)$ denotes the number of nonzero terms whose coefficients are not $1 \in \mathbf{F}_q$.

For the code C_Γ in Eq. (4), define the divisor $D = P_1 + \cdots + P_n$. Define $\mathcal{L}(-G + \infty Q) = \bigcup_{i=1}^{\infty} \mathcal{L}(-G + iQ)$ for a positive divisor G of $F/\mathbf{F}_q$. Then $\mathcal{L}(-D + \infty Q)$ is an ideal of $\mathcal{L}(\infty Q)$ [33]. Let η_i be any element in $\mathcal{L}(-D + \infty Q)$ such that $\textsc{lm}(\eta_i) = x_1^j y_i$ with j being the minimal given i. Then by [31, Proposition 1], $\{\eta_0, \ldots, \eta_{a_1-1}\}$ is a Gröbner basis for $\mathcal{L}(-D + \infty Q)$ with respect to $<_s$ as an $\mathbf{F}_q[x_1]$-module. For a nonnegative integer s, define $\Gamma^{(\le s)} = \{s' \in \Gamma \mid s' \le s\}$, $\Gamma^{(>s)} = \{s' \in \Gamma \mid s' > s\}$, and $\mathrm{prec}(s) = \max\{s' \in H(Q) \mid s' < s\}$. We define $\mathrm{prec}(0) = -1$.

3.3 Precompuation before Getting a Received Word

Before getting $\vec{r}$, we need to compute the Pellikaan-Miura standard form of the algebraic curve, $y_0 (= 1), y_1, \ldots, y_{a_1-1}$, and φ_s for $s \in H(Q)$ as defined in Section 2. Also compute $\eta_0, \ldots, \eta_{a_1-1}$, which can be done by [33].

For each (i, j), express $y_i y_j$ as an $\mathbf{F}_q$-linear combination of monomials in Ω_0. Such expressions will be used for computing products and quotients in $\mathcal{L}(\infty Q)$ as explained in Section 3.4.1. From the above data, we can easily know $\textsc{lc}(y_i y_j)$, which will be used in Eqs. (14) and (22).

Find elements $\varphi_s \in \Omega_0$ with $s \in \widehat{H}(Q)$. There are n such elements, which we denote by $\psi_1, \ldots \psi_n$ such that $-v_Q(\psi_i) < -v_Q(\psi_{i+1})$. Compute the $n \times n$ matrix

$$M = \begin{pmatrix} \psi_1(P_1) & \cdots & \psi_1(P_n) \\ \vdots & \vdots & \vdots \\ \psi_n(P_1) & \cdots & \psi_n(P_n) \end{pmatrix}^{-1}. \tag{7}$$

3.4 Multiplication and Division in an Affine Coordinate Ring

In both original unique decoding algorithm [31] and our modified version, we need to quickly compute the product gh of two elements g, h in the affine coordinate ring $\mathcal{L}(\infty Q)$. In our modified version, we also need to compute the quotient g/h depending on the choice of iteration termination criterion described in Section 3.6. Since the authors could not find quick computational procedures for those tasks in $\mathcal{L}(\infty Q)$, we shall present such ones here.

3.4.1 Multiplication in an Affine Coordinate Ring

The normal form of g, for $g \in \mathcal{L}(\infty Q)$, is the expression of g written as an $\mathbf{F}_q$-linear combination of monomials $\varphi_s \in \Omega_0$. g, h are assumed to be in the normal form. We propose the following procedure to compute the normal form of gh. Let the normal form of $y_i y_j$ be

$$\sum_{k=0}^{a_1-1} y_k f_{i,j,k}(x_1).$$

with $f_{i,j,k}(x_1) \in \mathbf{F}_q[x_1]$, which is computed in Section 3.3.

We denote by $X_1, Y_1, \ldots, Y_{a_1-1}$ algebraically independent variables over $\mathbf{F}_q$.

1. Assume that g and h are in their normal forms. Change y_i to Y_i and x_1 to X_1 in g, h for $i = 1, \ldots, a_1 - 1$. Recall that $y_0 = 1$. Denote the results by G, H.

2. Compute GH. This step needs

$$\gamma(g) \times \gamma(h) \tag{8}$$

multiplications in $\mathbf{F}_q$.

3. Let $GH = \sum_{0 \le i,j < a_1} Y_i Y_j F_{G,H,i,j}(X_1)$. Then we have

$$gh = \sum_{0 \le i,j < a_1} F_{G,H,i,j}(x_1) \sum_{k=0}^{a_1-1} y_k f_{i,j,k}(x_1). \tag{9}$$

Computation of $F_{G,H,i,j}(X_1) \sum_{k=0}^{a_1-1} y_k f_{i,j,k}(x_1)$ needs at most $\gamma_{\neq 1}(\sum_{k=0}^{a_1-1} y_k f_{i,j,k}(x_1))$ $\gamma(F_{G,H,i,j}(x_1))$ multiplications in $\mathbf{F}_q$. Therefore, the total number of multiplications in $\mathbf{F}_q$ in this step is at most

$$\sum_{0 \le i,j < a_1} \gamma(F_{G,H,i,j}(x_1))\gamma_{\neq 1}(\sum_{k=0}^{a_1-1} y_k f_{i,j,k}(x_1)). \tag{10}$$

Therefore, the total number of multiplications in $\mathbf{F}_q$ is at most

$$\gamma(g) \times \gamma(h) + \sum_{0 \le i,j < a_1} \gamma(F_{G,H,i,j}(x_1))\gamma_{\neq 1}(\sum_{k=0}^{a_1-1} y_k f_{i,j,k}(x_1)). \tag{11}$$

Define Eq. (11) as multi(g, h).

We emphasize that when the characteristic of $\mathbf{F}_q$ is 2 and all the coefficients of defining equations belong to $\mathbf{F}_2$, which is almost always the case for those cases of interest for applications in coding theory, then $\gamma_{\neq 1}(\sum_{k=0}^{a_1-1} y_k f_{i,j,k}(x_1))$ in Eq. (11) is zero. This means that $\mathcal{L}(\infty Q)$ has little additional overhead over $\mathbf{F}_q[X]$ for computing products of their elements in terms of the number of $\mathbf{F}_q$-multiplications and divisions.

Remark 4 *Define (i, j) to be equivalent to (i', j') if $y_i y_j = y_{i'} y_{j'} \in \mathcal{L}(\infty Q)$. Denote by $[i, j]$ the equivalence class represented by (i, j). For $(i, j), (i', j') \in [i, j]$ we have $f_{i,j,k}(x_1) = f_{i',j',k}(x_1)$, which is denoted by $f_{[i,j],k}(x_1)$. The right hand side of Eq. (9) can be written as*

$$\sum_{[i,j]} \left(\sum_{(i',j') \in [i,j]} F_{G,H,i',j'}(x_1) \right) \sum_{k=0}^{a_1-1} y_k f_{[i,j],k}(x_1). \tag{12}$$

By using Eq. (12) instead of Eq. (9), we have another upper bound on the number of multiplications as

$$\gamma(g) \times \gamma(h) + \sum_{[i,j]} \gamma \left(\sum_{(i',j') \in [i,j]} F_{G,H,i',j'}(x_1) \right) \gamma_{\neq 1}(\sum_{k=0}^{a_1-1} y_k f_{[i,j],k}(x_1)). \tag{13}$$

Since

$$\gamma\left(\sum_{(i',j')\in[i,j]} F_{G,H,i',j'}(x_1)\right) \le \sum_{(i',j')\in[i,j]} \gamma(F_{G,H,i',j'}(x_1)),$$

we have Eq. (13) ≤ Eq. (11). However, Eq. (13) is almost always the same as Eq. (11) over the curve in Example 2, and Eq. (13) will not be used in our computer experiments in Section 5.

3.4.2 Computation of the Quotient

Assume $h \neq 0$. The following procedure computes the quotient $g/h \in \mathcal{L}(\infty Q)$ or declares that g does not belong to the principal ideal of $\mathcal{L}(\infty Q)$ generated by h.

1. Initialize $\sigma = 0$. Also initialize $\zeta = 0$.

2. Check if $-v_Q(g) \in -v_Q(h) + H(Q)$. If not, declare that g does not belong to the principal ideal of $\mathcal{L}(\infty Q)$ generated by h, and finish the procedure.

3. Let $\varphi_s \in \Omega_0$ such that $-v_Q(g) = -v_Q(\varphi_s h)$. Observe that $\textsc{lc}(\varphi_s \textsc{lm}(h)) = \textsc{lc}(y_{s \bmod a_1} y_{-v_Q(h) \bmod a_1})$ and that $\textsc{lc}(y_{s \bmod a_1} y_{-v_Q(h) \bmod a_1})$ is precomputed as Section 3.3. Let

$$\mathbf{F}_q \ni t = \textsc{lc}(g)/(\textsc{lc}(h) \times \underbrace{\textsc{lc}(\varphi_s \textsc{lm}(h))}_{\text{Precomputed in Section 3.3}}). \tag{14}$$

   Computation of $t\varphi_s$ needs one multiplication and one division in $\mathbf{F}_q$. Observe that $-v_Q(g - t\varphi_s h) < -v_Q(g)$.

4. Compute the normal form of $t\varphi_s h$, which requires at most $\mathrm{multi}(t\varphi_s, h)$ multiplications in $\mathbf{F}_q$. Increment ζ by $2 + \mathrm{multi}(t\varphi_s, h)$.

5. Update $\sigma \leftarrow \sigma + t\varphi_s$ and $g \leftarrow g - t\varphi_s h$. If the updated g is zero, then output the updated σ as the quotient and finish the procedure. Otherwise go to Step 2. This step has no multiplication nor division.

Define $\mathrm{quot}(g, h)$ as ζ after finishing the above procedure. $\mathrm{quot}(g, h)$ is an upper bound on the number of multiplications and divisions in $\mathbf{F}_q$ in the above procedure. The program variable ζ is just to define $\mathrm{quot}(g, h)$, and the decoding algorithm does not need to update ζ. Observe also that the above procedure is a straightforward generalization of the standard long division of two univariate polynomials [36].

3.5 Initialization after Getting a Received Word $\vec{r}$

Let $(i_1, \ldots, i_n)^T = M\vec{r}$, where M is defined in Eq. (7). Define $h_{\vec{r}} = \sum_{j=1}^{n} i_j \psi_j$. Then we have $\mathrm{ev}(h_{\vec{r}}) = \vec{r}$. The computation of $h_{\vec{r}}$ from $\vec{r}$ needs at most n^2 multiplications in $\mathbf{F}_q$.

Let $N = -v_Q(h_{\vec{r}})$. For $i = 0, \ldots, a_1 - 1$, compute $g_i^{(N)} = \eta_i \in \mathcal{L}(\infty Q)$ and $f_i^{(N)} = y_i(z - h_{\vec{r}}) \in \mathcal{L}(\infty Q)z \oplus \mathcal{L}(\infty Q)$. The computation of $f_i^{(N)}$ needs at most $\mathrm{multi}(y_i, h_{\vec{r}})$ multiplications in $\mathbf{F}_q$. Therefore, the total number of multiplications in the initialization is at most

$$n^2 + \sum_{i=0}^{a_1-1} \mathrm{multi}(y_i, h_{\vec{r}}). \tag{15}$$

Let $s = N$ and execute the following steps.

3.6 Three Termination Criteria of the Iteration

After finishing the initialization step in Section 3.5, we iteratively compute $f_i^{(s)}$ and $g_i^{(s)}$ with $N \geq s \in H(Q) \cup \{-1\}$ and w_s with $N \geq s \in H(Q)$ from larger s to smaller s. The single iteration consists of two parts: The first part is to check if an iteration termination criterion is satisfied. The second part is computation of $f_i^{(s)}$ and $g_i^{(s)}$ for $N \geq s \in H(Q) \cup \{-1\}$. We describe the first part in Section 3.6.

Let $f_{\min} = \alpha_0 + z\alpha_1$ having the smallest $-v_Q(\alpha_1)$ among $f_0^{(s)}, \ldots, f_{a_1-1}^{(s)}$. In the following subsections, we shall propose three different procedures to judge whether or not iterations in the proposed algorithm can be terminated. In an actual implementation of the proposed algorithm, one criterion is chosen and the chosen one is consistently used throughout the iterations. The first one and the second one are different generalizations of [2, Theorem 12] for the case $g = 0$ to $g > 0$. Ali and Kuijper [2, Theorem 12] proved that if the number δ of errors satisfies $2\delta < d_{\mathrm{RS}}(C_s)$, where $d_{\mathrm{RS}}(C_s)$ is the minimum distance $n - s$ of the $[n, s + 1]$ Reed-Solomon code C_s, then the transmitted information word is obtained by Ali-Kuijper's algorithm as $-\alpha_0/\alpha_1$. To one-point primal AG codes, $d_{\mathrm{RS}}(C_s)$ can be generalized as either $d_{\mathrm{AG}}(C_s)$ or $n - s - g$. The former generalization $d_{\mathrm{AG}}(C_s)$ corresponds to the first criterion in Section 3.6.1 and the latter $n-s-g$ corresponds to the second in Section 3.6.2.

The third one is almost the same as the original procedure in [31]. The first one was proposed in [22] while the second and the third ones are new in this paper. We shall compare the three criteria in Section 5.2. Throughout this paper, $\mathrm{wt}(\vec{x})$ denotes the Hamming weight of a vector $\vec{x} \in \mathbf{F}_q^n$.

3.6.1 First Criterion for Judging Termination

If

- $s \in \Gamma$,
- $d_{AG}(C_{\Gamma^{(\leq s)}}) > 2\tau$, and
- $-v_Q(\alpha_1) \leq \tau + g$

then do the following:

1. Compute $\alpha_0/\alpha_1 \in F$. This needs at most

$$\mathrm{quot}(\alpha_0, \alpha_1) \tag{16}$$

   multiplications and divisions in $\mathbf{F}_q$.

2. If $\alpha_0/\alpha_1 \in \mathcal{L}(\infty Q)$ and α_0/α_1 can be written as a linear combination of monomials in $\{\varphi_{s'} \in s' \in \Gamma^{(\leq s)}\}$, then do the following:

   (a) If $d_{AG}(C_\Gamma) > 2\tau$ or $-v_Q(\alpha_1) \leq \tau$ then include the coefficients of $-\alpha_0/\alpha_1 + \sum_{s' \in \Gamma^{(>s)}} w_{s'}\varphi_{s'}$ into the list of transmitted information vectors, and avoid proceeding with the rest of the decoding procedure.

   (b) Otherwise compute $\mathrm{ev}(-\alpha_0/\alpha_1 + \sum_{s' \in \Gamma^{(>s)}} w_{s'}\varphi_{s'})$. This needs at most

$$n\gamma(-\alpha_0/\alpha_1 + \sum_{s' \in \Gamma^{(>s)}} w_{s'}\varphi_{s'}) \tag{17}$$

   multiplications and divisions in $\mathbf{F}_q$.

   (c) If

$$\mathrm{wt}\left(\mathrm{ev}(-\alpha_0/\alpha_1 + \sum_{s' \in \Gamma^{(>s)}} w_{s'}\varphi_{s'}) - \vec{r}\right) \leq \tau,$$

   then include the coefficients of $-\alpha_0/\alpha_1 + \sum_{s' \in \Gamma^{(>s)}} w_{s'}\varphi_{s'}$ into the list of transmitted information vectors, and avoid proceeding with s. Otherwise, continue the iterations unless $s < n - g - 2\tau$.

3.6.2 Second Criterion for Judging Termination

If $s = \max\{s' \in \Gamma \mid s' < n - 2\tau - g\}$, then do the following:

1. If $-v_Q(\alpha_1) > \tau + g$ then stop proceeding with iteration.

2. Otherwise compute $\alpha_0/\alpha_1 \in F$. This needs at most

$$\mathrm{quot}(\alpha_0, \alpha_1) \tag{18}$$

   multiplications and divisions in $\mathbf{F}_q$.

3. If $2\tau < d_{\mathrm{AG}}(C_\Gamma)$, $\alpha_0/\alpha_1 \in \mathcal{L}(\infty Q)$, and α_0/α_1 can be written as a linear combination of monomials in $\{\varphi_{s'} \in s' \in \Gamma^{(\leq s)}\}$ then declare the coefficients of $-\alpha_0/\alpha_1 + \sum_{s' \in \Gamma^{(>s)}} w_{s'}\varphi_{s'}$ as the only transmitted information and finish. Otherwise declare "decoding failure" and finish.

4. If $2\tau \geq d_{\mathrm{AG}}(C_\Gamma)$, $\alpha_0/\alpha_1 \in \mathcal{L}(\infty Q)$ and α_0/α_1 can be written as a linear combination of monomials in $\{\varphi_{s'} \in s' \in \Gamma^{(\leq s)}\}$, then do the following:

   (a) If $-v_Q(\alpha_1) \leq \tau$ then include the coefficients of $-\alpha_0/\alpha_1 + \sum_{s' \in \Gamma^{(>s)}} w_{s'}\varphi_{s'}$ into the list of transmitted information vectors, and avoid proceeding with s.

   (b) Otherwise compute $\mathrm{ev}(-\alpha_0/\alpha_1 + \sum_{s' \in \Gamma^{(>s)}} w_{s'}\varphi_{s'})$. This needs at most

$$n\gamma(-\alpha_0/\alpha_1 + \sum_{s' \in \Gamma^{(>s)}} w_{s'}\varphi_{s'}) \tag{19}$$

   multiplications and divisions in $\mathbf{F}_q$.

   (c) If

$$\mathrm{wt}\left(\mathrm{ev}(-\alpha_0/\alpha_1 + \sum_{s' \in \Gamma^{(>s)}} w_{s'}\varphi_{s'}) - \vec{r}\right) \leq \tau,$$

   then include the coefficients of $-\alpha_0/\alpha_1 + \sum_{s' \in \Gamma^{(>s)}} w_{s'}\varphi_{s'}$ into the list of transmitted information vectors.

5. Finish the iteration no matter what happened in the above steps.

3.6.3 Third Criterion for Judging Termination

Just repeat the iteration until finding $f_i^{(s)}$ at $s = -1$ for $i = 0, \dots, a_1 - 1$. If $2\tau < d_{\mathrm{AG}}(C_\Gamma)$ then declare the vector $(w_s : s \in \Gamma)$ as the only transmitted information and finish.

If $2\tau \geq d_{\mathrm{AG}}(C_\Gamma)$ then do the following:

1. If $\alpha_0 = 0$ and $-v_Q(\alpha_1) \leq \tau$ then include the vector $(w_s : s \in \Gamma)$ into the list of transmitted information vectors. Finish the iteration.

2. If $-v_Q(\alpha_1) > \tau + g$ then finish the iteration.

3. Otherwise compute $\mathrm{ev}(\sum_{s\in\Gamma} w_s\varphi_s)$. This needs at most

$$n\gamma(\sum_{s\in\Gamma} w_s\varphi_s) \tag{20}$$

   multiplications and divisions in $\mathbf{F}_q$.

4. If

$$\mathrm{wt}\left(\mathrm{ev}(\sum_{s\in\Gamma} w_s\varphi_s) - \vec{r}\right) \leq \tau,$$

   then include the vector $(w_s : s \in \Gamma)$ into the list of transmitted information vectors. Finish the iteration.

3.7 Iteration of Pairing, Voting, and Rebasing

The iteration of the original algorithm [31] consists of three steps, called pairing, voting, and rebasing. We will make a small change to the original. Our modified version is described below.

3.7.1 Pairing

Let

$$g_i^{(s)} = \sum_{0\leq j<a_1} c_{i,j} y_j z + \sum_{0\leq j<a_1} d_{i,j} y_j, \text{ with } c_{i,j}, d_{i,j} \in \mathbf{F}_q[x_1],$$
$$f_i^{(s)} = \sum_{0\leq j<a_1} a_{i,j} y_j z + \sum_{0\leq j<a_1} b_{i,j} y_j, \text{ with } a_{i,j}, b_{i,j} \in \mathbf{F}_q[x_1],$$

and let $\nu_i^{(s)} = \textsc{lc}(d_{i,i})$. We assume that $\textsc{lt}(f_i^{(s)}) = a_{i,i}y_iz$ and $\textsc{lt}(g_i^{(s)}) = d_{i,i}y_i$. For $0 \le i < a_1$, as in [31] there are unique integers $0 \le i' < a_1$ and k_i satisfying

$$-v_Q(a_{i,i}y_i) + s = a_1k_i - v_Q(y_{i'}).$$

Note that by the definition above

$$i' = i + s \bmod a_1, \tag{21}$$

and the integer $-v_Q(a_{i,i}y_i) + s$ is a nongap if and only if $k_i \ge 0$. Now let $c_i = \deg_{x_1}(d_{i',i'}) - k_i$. Note that the map $i \mapsto i'$ is a permutation of $\{0, 1, \ldots, a-1\}$ and that the integer c_i is defined such that $a_1c_i = -v_Q(d_{i',i'}y_{i'}) + v_Q(a_{i,i}y_i) - s$.

3.7.2 Voting

For each $i \in \{0, \ldots a_1 - 1\}$, we set

$$\mu_i = \textsc{lc}(a_{i,i}y_i\varphi_s),\ w_{s,i} = -\frac{b_{i,i'}[x_1^{k_i}]}{\mu_i},\ \overline{c_i} = \max\{c_i, 0\}, \tag{22}$$

where $b_{i,i'}[x_1^{k_i}]$ denotes the coefficient of x_{k_i} of the univariate polynomial $b_{i,i'} \in \mathbf{F}_q[x_1]$. We remark that the leading coefficient μ_i must be considered after expressing $a_{i,i}y_i\varphi_s$ by monomials in Ω_0.

Observe that $\textsc{lc}(y_i\varphi_s) = \textsc{lc}(y_iy_{s \bmod a_1})$ and that $\textsc{lc}(y_iy_{s \bmod a_1})$ is already precomputed as Section 3.3. By using that precomputed table, computation of μ_i needs one multiplication. The total number of multiplications and divisions in Eq. (22) is

$$2a_1 \tag{23}$$

excluding negation from the number of multiplication.

Let

$$\nu(s) = \frac{1}{a_1}\sum_{0\le i<a_1} \max\{-v_Q(\eta_{i'}) + v_Q(y_i) - s, 0\}. \tag{24}$$

We consider two different candidates depending on whether $s \in \Gamma$ or not:

- If $s \in H(Q) \setminus \Gamma$, set
$$w = 0. \tag{25}$$

- If $s \in \Gamma$, let w be one of the element(s) in $\mathbf{F}_q$ with
$$\sum_{w=w_{s,i}} \overline{c_i} \ge \sum_{w\ne w_{s,i}} \overline{c_i} - 2\tau + \nu(s). \tag{26}$$

Let $w_s = w$. If several w's satisfy the condition above, repeat the rest of the algorithm for each of them.

3.7.3 Rebasing

In all of the following cases, we need to compute the normal form of the product $w\varphi_s \times \sum_{j=0}^{a_1-1} a_{i,j}y_j$, and the product $w\varphi_s \times \sum_{j=0}^{a_1-1} c_{i,j}y_j$. For each i, the number of multiplications is

$$\leq \text{multi}(w\varphi_s, \sum_{j=0}^{a_1-1} a_{i,j}y_j) + \text{multi}(w\varphi_s, \sum_{j=0}^{a_1-1} c_{i,j}y_j), \tag{27}$$

where multi$(\cdot,\cdot)$ is defined in Section 3.4.1.

- If $w_{s,i} = w$, then let

$$g_{i'}^{(\text{prec}(s))} = g_{i'}^{(s)}(z + w\varphi_s),$$
$$f_i^{(\text{prec}(s))} = f_i^{(s)}(z + w\varphi_s)$$

  where the parentheses denote substitution of the variable z and let $\nu_{i'}^{(\text{prec}(s))} = \nu_{i'}^{(s)}$. The number of multiplications in this case is bounded by Eq. (27).

- If $w_{s,i} \neq w$ and $c_i > 0$, then let

$$g_{i'}^{(\text{prec}(s))} = f_i^{(s)}(z + w\varphi_s),$$
$$f_i^{(\text{prec}(s))} = x_1^{c_i} f_i^{(s)}(z + w\varphi_s) - \frac{\mu_i(w - w_{s,i})}{\nu_{i'}^{(s)}} g_{i'}^{(s)}(z + w\varphi_s)$$

  and let $\nu_{i'}^{(\text{prec}(s))} = \mu_i(w - w_{s,i})$.
  Computation of $\frac{\mu_i(w-w_{s,i})}{\nu_{i'}^{(s)}}$ needs one multiplication and one division. The product of $\frac{\mu_i(w-w_{s,i})}{\nu_{i'}^{(s)}}$ and $g_{i'}^{(s)}(z + w\varphi_s)$ needs $\gamma(g_{i'}^{(s)}(z + w\varphi_s))$ multiplications, where γ is defined in Section 3.4.1. Thus, the number of multiplications and divisions is

$$\leq 2 + \gamma(g_{i'}^{(s)}(z + w\varphi_s)) + \text{Eq. (27)}. \tag{28}$$

- If $w_{s,i} \neq w$ and $c_i \leq 0$, then let

$$g_{i'}^{(\text{prec}(s))} = g_{i'}^{(s)}(z + w\varphi_s),$$
$$f_i^{(\text{prec}(s))} = f_i^{(s)}(z + w\varphi_s) - \frac{\mu_i(w - w_{s,i})}{\nu_{i'}^{(s)}} x_1^{-c_i} g_{i'}^{(s)}(z + w\varphi_s)$$

and let $\nu_{i'}^{(\mathrm{prec}(s))} = \nu_{i'}^{(s)}$.

Computation of $\frac{\mu_i(w-w_{s,i})}{\nu_{i'}^{(s)}}$ needs one multiplication and one division. The product of $\frac{\mu_i(w-w_{s,i})}{\nu_{i'}^{(s)}}$ and $g_{i'}^{(s)}(z + w\varphi_s)$ needs $\gamma(g_{i'}^{(s)}(z + w\varphi_s))$ multiplications, where γ is defined in Section 3.4.1. Thus, the number of multiplications and divisions is $\leq$ Eq. (28).

After computing $f_i^{(\mathrm{prec}(s))}$ and $g_i^{(\mathrm{prec}(s))}$ as above, update the program variable s to prec(s) and go to the beginning of Section 3.6.

3.8 Difference to the Original Method

In this subsection, we review advantages of our modified algorithm over the original [31].

- Our version can handle any one-point primal AG codes, while the original can handle codes only coming from the C_{ab} curves [37]. This generalization is enabled only by replacing y^j in [31] by y_j defined in Section 2.

- Our version can find all the codeword within Hamming distance τ from the received word $\vec{r}$, while the original is a unique decoding algorithm.

- Our version does not compute $f_i^{(s)}, g_i^{(s)}$ for a Weiestrass gap $s \notin H(Q)$, while the original computes them for $N \geq s \notin H(Q)$.

- The original algorithm assumed $u < n$, where u is as defined in Eq. (1). This assumption is replaced by another less restrictive assumption (5) in our version.

- Our version supports the Feng-Rao improved code construction [17], while the original does not. This extension is made possible by the change at Eq. (25).

- The first and the second termination criteria come from [2, Theorem 12] and do not exist in the original [31].

- The third termination criterion is essentially the same as the original [31], but examination of the Hamming distance between the decoded codeword and $\vec{r}$ is added when $2\tau \geq d_{\mathrm{AG}}(C_\Gamma)$.

- The original [31] is suitable for parallel implementation on electric circuit similar to the Kötter architecture [27]. Our modified version retains this advantage.

4 Theoretical Analysis of the Proposed Modification

In this section we prove that our modified algorithm can find all the codewords within Hamming distance τ from the received word $\vec{r}$. We also give upper bounds on the number of iterations in Section 4.6.

4.1 Supporting Lemmas

In Section 4.1 we shall introduce several lemmas necessary in Sections 4.2–4.5. Recall that the execution of our modified algorithm can branch when there are multiple candidates satisfying the condition (26). For a fixed sequence of determined w_s, define $\vec{r}^{(N)} = \vec{r}$ and recursively define $\vec{r}^{(\mathrm{prec}(s))} = \vec{r}^{(s)} - \mathrm{ev}(w_s\varphi_s)$. By definition $\vec{r}^{(-1)} = \vec{r} - \mathrm{ev}(\sum_{s\in\Gamma} w_s\varphi_s)$.

The following lemma explains why the authors include "Gröbner bases" in the paper title. The module $I_{\vec{r}^{(N)}}$ was used in [5, 18, 28, 30, 35, 42, 43] but the use of $I_{\vec{r}^{(s)}}$ with $s < N$ was new in [31].

Lemma 5 *Fix $s \in H(Q) \cup \{-1\}$. Let $\vec{r}^{(s)}$ correspond to w_s ($s \in \Gamma$) chosen by the decoding algorithm. Define the $\mathbf{F}_q[x_1]$-submodule $I_{\vec{r}^{(s)}}$ of $\mathcal{L}(\infty Q)z \oplus \mathcal{L}(\infty Q)$ by*

$$I_{\vec{r}^{(s)}} = \{\alpha_0 + \alpha_1 z \mid \alpha_0, \alpha_1 \in \mathcal{L}(\infty Q), v_{P_i}(\alpha_0 + r_i^{(s)}\alpha_1) \geq 1, 1 \leq i \leq n\}, \tag{29}$$

where $\vec{r}^{(s)} = (r_1^{(s)}, \ldots, r_n^{(s)})$. Then $\{f_i^{(s)}, g_i^{(s)} \mid i = 0, \ldots, a_1 - 1\}$ is a Gröbner basis of $I_{\vec{r}^{(s)}}$ with respect to $<_s$ as an $\mathbf{F}_q[x_1]$-module.

Proof. This lemma is a generalization of [31, Proposition 11]. We can prove this lemma in exactly the same way as the proof of [31, Proposition 11] with replacing y^j in [31] with y_j and $s - 1$ in [31] by prec(s). ■

The following proposition shows that the original decoding algorithm [31] can correct errors up to half the bound $d_{\mathrm{AG}}(C_\Gamma)$, which was not claimed in [31].

Proposition 6 *Fix $s \in \Gamma$. Let $\lambda(s)$ as defined in Eq. (6) and $\nu(s)$ as defined in Eq. (24). Then $\nu(s) = \lambda(s)$.*

Proof. Let $T_i = \{j \in H(Q) \mid j \equiv i \pmod{a_1}, j + s \in \widehat{H}(Q)\}$, then we have $\lambda(s) = \sharp T_0 + \cdots + \sharp T_{a_1-1}$. Moreover, observe that

$$H(Q) \setminus \widehat{H}(Q) = \{-v_Q(\eta_i x_1^k) \mid i = 0, \ldots, a_1 - 1, k = 0, 1, \ldots\}.$$

Therefore, for $s \in \Gamma$ we have

$$\begin{aligned}
T_i &= \{j \in H(Q) \mid j \equiv i \pmod{a_1}, j + s \in \widehat{H}(Q)\} \\
&= \{j \in H(Q) \mid j \equiv i \pmod{a_1}, j + s \notin H(Q) \setminus \widehat{H}(Q)\} \\
&= \{j \in H(Q) \mid j \equiv i \pmod{a_1}, j + s \notin \{-v_Q(\eta_{i'} x_1^k) \mid k \geq 0\}\} \\
&= \{-v_Q(y_i x_1^m) \mid s - v_Q(y_i x_1^m) \notin \{-v_Q(\eta_{i'} x_1^k) \mid k \geq 0\}\},
\end{aligned}$$

where the third equality holds by Eq. (21). By the equalities above, we see

$$\sharp T_i = \max\left\{0, \frac{-v_Q(\eta_{i'}) + v_Q(y_i) - s}{-v_Q(x_1)}\right\},$$

which proves the equality $\nu(s) = \lambda(s)$. ∎

Lee et al. [31] showed that their original decoding algorithm can correct up to $\lfloor (d_{\mathrm{LBAO}}(C_u) - 1)/2 \rfloor$ errors, where $d_{\mathrm{LBAO}}(C_u) = \min\{\nu(s) \mid s \in H(Q), s \leq u\}$. Proposition 6 implies that $d_{\mathrm{LBAO}}(C_u)$ is equivalent to $d_{\mathrm{AG}}(C_u)$ for every one-point primal code C_u, and therefore [3, Theorem 8] implies [31, Proposition 12]. In another recent paper [23] we proved that d_{AG} and d_{LBAO} are equal to the Feng-Rao bound as defined in [6, 32] for C_u.

4.2 Lower Bound for the Number of Votes

In Section 4.2 we discuss the number of votes (26) which a candidate $w_{s,i}$ receives. Since we study list decoding, we cannot assume the original transmitted codeword nor the error vector as in [31]. Nevertheless, the original theorems in [31] allow natural generalizations to the list decoding context.

Lemma 7 *Fix $s \in \Gamma$. For $s' \in \Gamma^{(>s)}$, fix a sequence of $w_{s'}$ chosen by the decoding algorithm, and define $\vec{r}^{(s)}$ corresponding to the chosen sequence of $w_{s'}$. Fix $\omega_s \in \mathbf{F}_q$. Let $\vec{e} = (e_1, \ldots, e_n)^T$ be a nonzero vector with the minimum Hamming weight in the coset $\vec{r}^{(s)} - \mathrm{ev}(\omega_s \varphi_s) + C_{s-1}$, where C_{s-1} is as defined in Eq. (1). Define*

$$\begin{aligned}
J_{\vec{e}} &= \bigcap_{e_i \neq 0} \mathcal{L}(-P_i + \infty Q) \\
&= \mathcal{L}\left(\infty Q - \sum_{e_i \neq 0} P_i\right) \text{ (by [33]).}
\end{aligned}$$

Let $\{\epsilon_0, \ldots, \epsilon_{a_1-1}\}$ *be a Gröbner basis for* $J_{\vec{e}}$ *as an* $\mathbf{F}_q[x_1]$*-module with respect to* $<_s$ *(for any integer* s*), such that* $\textsc{lm}(\epsilon_j) = x_1^{k_j} y_j$*.*

Under the above notations, we have

$$\begin{aligned} -v_Q(\epsilon_i) + v_Q(a_{i,i}y_i) &\geq a_1\overline{c_i}, \\ \min\{-v_Q(\epsilon_i) + s, -v_Q(\eta_{i'})\} &\geq -v_Q(d_{i',i'}y_{i'}), \end{aligned}$$

for i *with* $w_{s,i} \neq \omega_s$*, and*

$$\min\{-v_Q(\epsilon_i) + s, -v_Q(\eta_{i'})\} \geq -v_Q(d_{i',i'}y_{i'}) - a_1\overline{c_i},$$

for i *with* $w_{s,i} = \omega_s$*.*

Proof. The proof is the same as those of [31, Propositions 7 and 8], with replacing y^j in [31] by y_j, $\delta(\cdot)$ in [31] by $-v_Q(\cdot)$. ■

The following lemma is a modification to [31, Proposition 9] for the list decoding.

Lemma 8 *We retain notations from Lemma 7. We have*

$$\begin{aligned} a_1 \sum_{w_{s,i}=\omega_s} \overline{c_i} \geq a_1 \sum_{w_{s,i}\neq\omega_s} \overline{c_i} - 2a_1 \mathrm{wt}(\vec{e}) \\ + \sum_{0\leq i<a_1} \max\{-v_Q(\eta_{i'}) + v_Q(y_i) - s, -v_Q(\epsilon_i) + v_Q(y_i)\}. \end{aligned}$$

Proof. Lemma 7 implies

$$\begin{aligned} \sum_{w_{s,i}=\omega_s} a_1\overline{c_i} &\geq \sum_{w_{s,i}=\omega_s} -v_Q(d_{i',i'}y_{i'}) - \min\{-v_Q(\epsilon_i) + s, -v_Q(\eta_{i'})\} \\ &\geq \sum_{0\leq i<a_1} -v_Q(d_{i',i'}y_{i'}) - \min\{-v_Q(\epsilon_i) + s, -v_Q(\eta_{i'})\} \end{aligned}$$

and

$$\begin{aligned} \sum_{w_{s,i}\neq\omega_s} a_1\overline{c_i} &\leq \sum_{w_{s,i}\neq\omega_s} -v_Q(\epsilon_i) + v_Q(a_{i,i}y_i) \\ &\leq \sum_{0\leq i<a_1} -v_Q(\epsilon_i) + v_Q(a_{i,i}y_i). \end{aligned}$$

Now we have a chain of inequalities

$$
\begin{aligned}
&\sum_{w_{s,i}=\omega_s} a_1\overline{c_i} - \sum_{w_{s,i}\neq\omega_s} a_1\overline{c_i} \\
\geq\ & \sum_{0\leq i<a_1} -v_Q(d_{i',i'}y_{i'}) - \min\{-v_Q(\epsilon_i)+s, -v_Q(\eta_{i'})\} \\
& - \sum_{0\leq i<a_1} -v_Q(\epsilon_i) + v_Q(a_{i,i}y_i) \\
=\ & \sum_{0\leq i<a_1} -v_Q(d_{i',i'}y_{i'}) - v_Q(a_{i,i}y_i) \\
& - \min\{-v_Q(\epsilon_i)+s, -v_Q(\eta_{i'})\} + v_Q(\epsilon_i) \\
=\ & \sum_{0\leq i<a_1} -v_Q(\eta_{i'}) - v_Q(y_i) \qquad (30) \\
& + \max\{+v_Q(\epsilon_i)-s, +v_Q(\eta_{i'})\} + v_Q(\epsilon_i) \\
=\ & \sum_{0\leq i<a_1} \max\{-v_Q(\eta_{i'}) + v_Q(y_i) - s, -v_Q(\epsilon_i) + v_Q(y_i)\} \\
& - \sum_{0\leq i<a_1} 2(-v_Q(\epsilon_i) + v_Q(y_i))
\end{aligned}
$$

where at Eq. (30) we used the equality

$$
\begin{aligned}
& \sum_{0\leq i<a_1} -v_Q(d_{i',i'}y_{i'}) + \sum_{0\leq i<a_1} -v_Q(a_{i,i}y_i) \\
=\ & \sum_{0\leq i<a_1} -v_Q(d_{i,i}y_i) + \sum_{0\leq i<a_1} -v_Q(a_{i,i}y_i) \\
=\ & \sum_{0\leq i<a_1} (-v_Q(d_{i,i}) - v_Q(a_{i,i})) + \sum_{0\leq i<a_1} -2v_Q(y_i) \\
=\ & a_1 n + \sum_{0\leq i<a_1} -2v_Q(y_i) \\
=\ & \sum_{0\leq i<a_1} (-v_Q(\eta_i) + v_Q(y_i)) + \sum_{0\leq i<a_1} -2v_Q(y_i) \\
=\ & \sum_{0\leq i<a_1} -v_Q(\eta_{i'}) + \sum_{0\leq i<a_1} -v_Q(y_i)
\end{aligned}
$$

shown in [31, Lemma 2 and Eq. (1)]. Finally note that

$$\sum_{0\leq i<a_1} 2(-v_Q(\epsilon_i) + v_Q(y_i))$$

$$= \sum_{0 \le i < a_1} 2a_1 \deg_{x_1}(\epsilon_i) = 2a_1 \mathrm{wt}(\vec{e})$$

by [31, Eq. (3)]. ■

The following lemma is a modification to [31, Proposition 10] for list decoding, and provides a lower bound for the number of votes (26) received by any candidate $\omega_s \in \mathbf{F}_q$, as indicated in the section title.

Proposition 9 *We retain notations from Lemma 7. Let $\nu(s)$ be as defined in Eq. (24). We have*

$$\sum_{w_{s,i}=\omega_s} \bar{c_i} \ge \sum_{w_{s,i}\neq\omega_s} \bar{c_i} - 2\mathrm{wt}(\vec{e}) + \nu(s).$$

Proof. We have

$$\begin{aligned} & \sum_{0 \le i < a_1} \max\{-v_Q(\eta_{i'}) + v_Q(y_i) - s, -v_Q(\epsilon_i) + v_Q(y_i)\} \\ \ge & \sum_{0 \le i < a_1} \max\{-v_Q(\eta_{i'}) + v_Q(y_i) - s, 0\} \end{aligned}$$

as $-v_Q(\epsilon_i) + v_Q(y_i) \ge 0$ for $0 \le i < a_1$. ■

4.3 Correctness of the Modified List Decoding Algorithm with the Third Iteration Termination Criterion

In this subsection and the following sections, we shall prove that the proposed list decoding algorithm will find all the codewords within the Hamming distance τ from the received word $\vec{r}$. Since the third iteration termination criterion is the easiest to analyze, we start with the third one.

Fix a sequence w_s for $s \in \Gamma$. If $\mathrm{wt}(\vec{r} - \mathrm{ev}(\sum_{s\in\Gamma} w_s\varphi_s)) \le \tau$ then the sequence w_s is found by the algorithm because of Proposition 9. When $2\tau < d_{\mathrm{AG}}(C_\Gamma)$, by Proposition 6 the decoding is not list decoding, and the algorithm just declares the sequence w_s as the transmitted information.

On the other hand, if $2\tau \ge d_{\mathrm{AG}}(C_\Gamma)$, then the found sequence could correspond to a codeword more distant than Hamming distance τ, and the algorithm examines the Hamming distance between the found codeword and the received word $\vec{r}$.

Since computing $\mathrm{ev}(f)$ for $f \in \mathcal{L}(\infty Q)$ needs many multiplications in $\mathbf{F}_q$, the algorithm checks some sufficient conditions to decide the Hamming distance between the found codeword and the received word $\vec{r}$. Let $\vec{r}^{(-1)} = (r_1^{(-1)}, \ldots, r_n^{(-1)})$. When $\alpha_0 = 0$ in Section 3.6.3, by Lemma 5, we have

$$\begin{aligned} \mathrm{wt}(\vec{r} - \mathrm{ev}(\sum_{s\in\Gamma} w_s\varphi_s)) &= \mathrm{wt}(\vec{r}^{(-1)}) \\ &\leq \sum_{r_i^{(-1)}\neq 0} v_{P_i}(\alpha_1) \\ &\leq -v_Q(\alpha_1), \end{aligned}$$

because Eq. (29) and $\alpha_0 = 0$ implies that $v_{P_i}(\alpha_1) \geq 1$ for $r_i^{(-1)} \neq 0$. By the above equation, $-v_Q(\alpha_1) \leq \tau$ implies that the found codeword is within Hamming distance τ from $\vec{r}$. This explains why the algorithm can avoid computation of the evaluation map ev in Step 1 in Section 3.6.3.

In order to explain Step 2 in Section 3.6.3, we shall show that the condition of Step 2 in Section 3.6.3 implies that $\mathrm{wt}(\vec{r} - \mathrm{ev}(\sum_{s\in\Gamma} w_s\varphi_s)) > \tau$. Suppose that $\mathrm{wt}(\vec{r} - \mathrm{ev}(\sum_{s\in\Gamma} w_s\varphi_s)) \leq \tau$. Then there exists $\beta_1 \in \mathcal{L}(\infty Q)$ such that $v_{P_i}(\beta) \geq 1$ for $r_i^{(-1)} \neq 0$, $-v_Q(\beta) \leq \tau + g$, and $\beta_1 z \in I_{\vec{r}^{(-1)}}$. Because the leading term of $\beta_1 z$ must be divisible by $\mathrm{LT}(f_i^{(-1)})$ for some i by the property of Gröbner bases, we must have $-v_Q(\alpha_1) \leq -v_Q(\beta_1)$. This explains why the algorithm can avoid computation of the evaluation map ev in Step 2 in Section 3.6.3.

Otherwise, the algorithm computes the Hamming distance between the found codeword and $\vec{r}$ in Steps 3 and 4 in Section 3.6.3.

4.4 Correctness of the Modified List Decoding Algorithm with the Second Iteration Termination Criterion

We shall explain why the second criterion in Section 3.6.2 correctly finds the required codewords. For explanation, we present slightly rephrased version of facts in [6].

Lemma 10 *[6, Lemma 2.3] Let $\beta_1 z + \beta_0 \in I_{\vec{r}^{(s)}}$ with $\mathrm{LT}(\beta_1 z + \beta_0) = \mathrm{LT}(\beta_1 z)$ with respect to $<_s$ and $-v_Q(\beta_1) < n - \tau - s$. If there exists $f \in \mathcal{L}(sQ)$ such that $\mathrm{wt}(\mathrm{ev}(f) - \vec{r}^{(s)}) \leq \tau$, then we have $f = -\beta_0/\beta_1$.*

Proof. Observe that $\mathrm{LT}(\beta_1 z + \beta_0) = \mathrm{LT}(\beta_1 z)$ implies that $-v_Q(\beta_0) \leq -v_Q(\beta_1) + s < n - \tau$. The claim of Lemma 10 is equivalent to [6, Lemma 2.3] with $A = (n-\tau-1)Q$ and $G = sQ$. Note that the assumption $\deg A > (n + \deg G)/2 + g - 1$ was not used in [6, Lemma 2.3] but only in [6, Lemma 2.4]. ∎ Note that the following proposition was essentially proved in [6, Proposition 2.10], [26, Section 14.2], and [48, Theorem 2.1] with $b = 1$.

Proposition 11 *Let α_0 and α_1 be as in Section 3.6.2. If $s < n - g - 2\tau$ and there exists $f \in \mathcal{L}(sQ)$ such that $\mathrm{wt}(\mathrm{ev}(f) - \vec{r}^{(s)}) \le \tau$, then we have $f = -\alpha_0/\alpha_1$.*

Proof. Let $g \in \mathcal{L}(\infty Q)$ such that $g(P_i) = 0$ if $f(P_i) \neq r_i^{(s)}$, and assume that g has the minimum pole order at Q among such elements in $\mathcal{L}(\infty Q)$. Then $-v_Q(g) \le \tau + g$. One has that $gz - fg \in I_{\vec{r}^{(s)}}$ and $\mathrm{LT}(gz - fg) = \mathrm{LT}(gz)$ with respect to $<_s$. By the property of Gröbner bases, $\mathrm{LT}(gz)$ is divisible by $\mathrm{LT}(f_i^{(s)})$ for some i, which implies $-v_Q(\alpha_1) \le -v_Q(g) \le \tau + g$. By Lemma 10 we have $f = -\alpha_0/\alpha_1$. ∎

We explain how the procedure in Section 3.6.2 works as desired. When the condition in Step 1 in Section 3.6.2 is true, then there cannot be a codeword within Hamming distance τ from $\vec{r}^{(s)}$ by the same reason as Section 4.3. So the algorithm stops processing with $\vec{r}^{(s)}$.

When $2\tau < d_{\mathrm{AG}}(C_\Gamma)$, then the algorithm declares $-\alpha_0/\alpha_1 + \sum_{s' \in \Gamma^{(>s)}} w_{s'}\varphi_{s'}$ as the unique codeword.

When $2\tau \ge d_{\mathrm{AG}}(C_\Gamma)$, then the algorithm examines the found codeword close enough to $\vec{r}$ in Steps 4a and 4b in Section 3.6.2. When $-v_Q(\alpha_1) \le \tau$ we can avoid computation of the evaluation map ev by the same reason as Section 4.3, which is checked at Step 4a. Otherwise we compute the codeword vector at Step 4b and examine its Hamming distance to $\vec{r}^{(s)}$.

By Proposition 11, the codeword must be found at $s = \max\{s' \in \Gamma \mid s' < n - 2\tau + g\}$. Therefore, we do not execute the iteration at $s < \max\{s' \in \Gamma \mid s' < n - 2\tau + g\}$.

4.5 Correctness of the Modified List Decoding Algorithm with the First Iteration Termination Criterion

We shall explain why the first criterion in Section 3.6.1 correctly finds the required codewords. The idea behind the first criterion is that there cannot be another codeword within Hamming distance τ from $\vec{r}^{(s)}$ when the algorithm already found one. So the algorithm can stop iteration with smaller s once a codeword is found as $\mathrm{ev}(-\alpha_0/\alpha_1 + \sum_{s' \in \Gamma^{(>s)}} w_{s'}\varphi_{s'})$.

The algorithm does not examine conditions when $-v_Q(\alpha_1) > \tau + g$ by the same reason as Sections 4.3 and 4.4. When $2\tau < d_{\mathrm{AG}}(C_\Gamma)$, then the algorithm declares $-\alpha_0/\alpha_1 + \sum_{s' \in \Gamma^{(>s)}} w_{s'}\varphi_{s'}$ as the unique codeword.

When $2\tau \ge d_{\mathrm{AG}}(C_\Gamma)$, then the algorithm examines the found codeword close enough to $\vec{r}$ in Steps 2a–2c in Section 3.6.1. When $-v_Q(\alpha_1) \le \tau$ we can avoid computation of the evaluation map ev by the same reason as Section 4.3, which is checked at Step 2a.

By Proposition 11, the codeword must be found at some $s \geq \max\{s' \in \Gamma \mid s' < n - 2\tau + g\}$. Therefore, we do not execute the iteration at $s < \max\{s' \in \Gamma \mid s' < n - 2\tau + g\}$.

4.6 Upper Bound on the Number of Iterations

Observe that for $s > \max\Gamma$ we set always w_s to 0. For each $s \in \Gamma$ satisfying $\nu(s) \leq 2\tau$, the number of accepted candidates satisfying Eq. (26) can be at most q. On the other hand, for s with $\nu(s) > 2\tau$, the number of candidates is either zero or one, because at most one $w \in \mathbf{F}_q$ can satisfy Eq. (26). Therefore, we have upper bounds for the number of iterations, counting executions of Rebasing in Section 3.7.3, as

$$\begin{aligned} &\sharp\{s \in H(Q) \mid \max\Gamma \leq s < N\} + \exp_q(\sharp\{s \in \Gamma \mid \nu(s) \leq 2\tau\}) \\ &\times\sharp\{s \in H(Q) \cup \{-1\} \mid s < \max\Gamma\} \end{aligned} \tag{31}$$

for the third criterion for judging termination, where $\exp_q(x) = q^x$, and

$$\begin{aligned} &\sharp\{s \in H(Q) \mid \max\Gamma \leq s < N)\} + \exp_q(\sharp\{s \in \Gamma \mid \nu(s) \leq 2\tau\}) \\ &\times\sharp\{s \in H(Q) \mid \max\{s' \in \Gamma \mid s' < n - 2\tau - g\} \leq s < \max\Gamma\} \end{aligned} \tag{32}$$

for the first and the second criteria for judging termination. We will use $\max \widehat{H}(Q)$ in place of N in Eqs. (31) and (32) for computation in Tables 1–4, because N depents on $\vec{r}$ and $N \leq \max \widehat{H}(Q)$.

Observe that the list decoding can be implemented as $\exp_q(\sharp\{s \in \Gamma \mid \nu(s) \leq 2\tau\})$ parallel execution of the unique decoding. Therefore, when one can afford $\exp_q(\sharp\{s \in \Gamma \mid \nu(s) \leq 2\tau\})$ parallel implementation, which increases the circuit size, the decoding time of list decoding is the same as that of the unique decoding.

5 Comparison to Conventional Methods

5.1 Simulation Condition and Results

We have provided an upper bound on the number of multiplications and divisions at each step of the proposed algorithm. We simulated 1,000 transmissions of codewords with the one-point primal codes on Klein quartic over $\mathbf{F}_8$ with $n = 23$

Table 1: Decoding results of codes on the Klein quartic ($\mathbf{F}_q = \mathbf{F}_8$, $g = 3$ and $n = 23$)

$\sharp\Gamma$	$d_{\mathrm{AG}}(C_\Gamma)$	# Errors $= \tau$	Termination Criterion in Sec. 3.6	# Iterations			# Multiplications & Divisions in $\mathbf{F}_q$		# Codewords Found	
				Eq.(31),(32)	Avg.	Max.	Avg.	Max.	Avg.	Max.
18	4	1	1st	11	8.00	8	1,170.09	1,254	1.00	1
			2nd	11	11.00	11	844.98	879		
			3rd	26	26.00	26	976.32	1,018		
		2	1st	328	196.63	260	26,203.96	77,209	1.34	3
			2nd	328	200.64	269	8,349.67	15,457		
			3rd	1,160	219.07	313	7,813.76	10,083		
		3	1st	28,680	11,996.34	13,353	1,626,658.69	2,490,386	19.75	28
			2nd	28,680	12,055.56	13,419	608,535.03	711,315		
			3rd	73,736	12,436.00	13,853	580,504.03	642,419		
11	10	4	1st	17	14.64	15	1,324.76	1,484	1.00	1
			2nd	17	16.64	17	1,161.52	1,293		
			3rd	26	25.64	26	1,329.07	1,468		
		5	1st	47	35.20	44	3,673.78	5,549	1.00	1
			2nd	47	38.20	47	2,915.04	3,622		
			3rd	103	45.41	72	3,072.08	3,769		
		6	1st	3,087	1,507.95	1,692	164,274.07	188,797	1.11	3
			2nd	3,087	1,511.28	1,695	113,592.10	130,810		
			3rd	5,647	1,535.23	1,725	113,472.30	130,697		

Table 2: Decoding results of the one-point Hermitian codes ($\mathbf{F}_q = \mathbf{F}_{16}$, $g = 6$, $n = 64$ and $\sharp\Gamma = 55$). The meanings of N and R in the third column is explained in Section 5.1.

$\sharp\Gamma$	$d_{AG}(C_\Gamma)$	# Errors $= \tau$	Termination Criterion in Sec. 3.6	# Iterations Eq.(31),(32)	# Iterations Avg.	# Iterations Max.	# Multiplications & Divisions in $\mathbf{F}_q$ Avg.	# Multiplications & Divisions in $\mathbf{F}_q$ Max.	# Codewords Found Avg.	# Codewords Found Max.
55	6	2R	1st	22	16.38	17	9,049.77	9,495	1.00	1
			2nd	22	21.79	22	5,483.91	5,614		
			3rd	70	69.79	70	6,331.16	6,530		
		2N	1st	22	16.22	17	9,005.92	9,477	1.00	1
			2nd	22	21.22	22	5,414.32	5,607		
			3rd	70	69.22	70	6,291.44	6,527		
		3R	1st	2,829	796.17	1,139	289,992.45	2,154,489	1.00	2
			2nd	2,829	800.66	1,143	116,784.32	164,299		
			3rd	14,605	846.78	1,191	117,848.30	130,366		
		3N	1st	2,829	750.73	817	334,588.68	360,413	2.28	5
			2nd	2,829	761.79	825	120,575.80	131,791		
			3rd	14,605	872.97	917	119,940.46	134,297		
		4R	1st	851,981	21,376.57	33,187	8,012,813.23	14,988,534	1.48	4
			2nd	851,981	21,384.19	33,198	2,431,318.50	3,763,057		
			3rd	3,735,565	21,458.60	33,327	2,432,782.60	3,761,206		
		4N	1st	851,981	21,744.53	32,943	10,952,709.73	16,938,498	4.29	5
			2nd	851,981	21,769.88	32,962	2,457,072.25	3,801,066		
			3rd	3,735,565	21,985.53	33,174	2,439,145.90	3,805,702		

Table 3: Decoding results of the one-point Hermitian codes ($\mathbf{F}_q = \mathbf{F}_{16}$, $g = 6$, $n = 64$ and $\sharp\Gamma = 39$). The meanings of N and R in the third column is explained in Section 5.1.

$\sharp\Gamma$	$d_{AG}(C_\Gamma)$	# Errors $= \tau$	Termination Criterion in Sec. 3.6	# Iterations			# Multiplications & Divisions in $\mathbf{F}_q$		# Codewords Found	
				Eq.(31),(32)	Avg.	Max.	Avg.	Max.	Avg.	Max.
39	20	9R	1st	36	30.77	31	10,726.51	11,175	1.00	1
			2nd	36	35.77	36	8,851.66	9,249		
			3rd	70	69.77	70	10,781.91	11,504		
		9N	1st	36	30.73	32	9,747.36	12,008	1.00	1
			2nd	36	35.73	36	7,807.55	8,378		
			3rd	70	69.73	70	9,607.34	10,645		
		10R	1st	143	74.99	93	37,643.43	47,179	1.00	1
			2nd	143	80.99	99	22,792.85	25,972		
			3rd	655	112.99	131	25,023.33	28,398		
		10N	1st	143	77.64	126	46,759.83	177,947	2.00	2
			2nd	143	89.61	138	23,971.51	43,332		
			3rd	655	153.73	244	28,063.19	36,453		
		11R	1st	36,895	12,112.08	12,859	7,480,228.09	7,911,646	1.00	1
			2nd	36,895	12,118.08	12,865	3,186,977.77	3,352,388		
			3rd	159,775	12,148.10	12,895	3,189,262.07	3,354,588		
		11N	1st	36,895	10,417.34	12,285	6,123,703.49	7,582,687	2.01	6
			2nd	36,895	10,429.34	12,297	2,638,130.92	3,226,308		
			3rd	159,775	10,491.11	12,357	2,641,014.19	3,230,078		

Table 4: Decoding results of codes on the curve in Example 2 ($\mathbf{F}_q = \mathbf{F}_9$, $g = 22$ and $n = 77$). We note that Eq. (31) give the same value for $\tau = 10$ and $\tau = 11$.

$\sharp\Gamma$	$d_{AG}(C_\Gamma)$	# Errors $= \tau$	Termination Criterion in Sec. 3.6	# Iterations			# Multiplications & Divisions in $\mathbf{F}_q$		# Codewords Found	
				Eq.(31),(32)	Avg.	Max.	Avg.	Max.	Avg.	Max.
58	6	2	1st	60	38.85	42	39,473.98	62,479	1.00	1
			2nd	60	59.30	60	12,255.73	13,348		
			3rd	89	88.30	89	13,710.71	14,886		
		3	1st	2,862	62.50	120	36,350.41	72,463	1.00	1
			2nd	2,862	79.62	134	21,754.75	30,030		
			3rd	5,049	106.62	161	23,556.44	31,555		
52	10	4	1st	64	46.94	51	37,228.61	78,090	1.00	1
			2nd	64	63.25	64	15,212.23	16,708		
			3rd	89	88.25	89	17,082.34	18,879		
		5	1st	196,866	48.96	163	24,776.23	88,893	1.00	1
			2nd	196,866	66.17	178	20,660.52	69,176		
			3rd	347,769	89.17	201	22,591.78	71,264		
37	20	9	1st	73	57.28	60	24,998.86	48,611	1.00	1
			2nd	73	72.34	73	23,168.98	24,784		
			3rd	89	88.34	89	25,655.70	27,675		
		10	1st	1,915	58.67	61	25,492.43	43,294	1.00	1
			2nd	1,915	74.39	75	27,355.54	29,452		
			3rd	3,049	88.39	89	29,678.00	31,836		
		11	1st	2,077	225.59	253	167,152.76	191,250	1.00	1
			2nd	2,077	242.36	268	124,664.66	139,519		
			3rd	3,049	254.36	280	126,693.96	141,989		

by using Examples 1 and 3, the one-point Hermitian codes over $\mathbf{F}_{16}$ with $n = 64$, and the one-point primal codes on the curve in Example 2 over $\mathbf{F}_9$ with $n = 77$.

The program is implemented on the Singular computer algebra system [9]. The program used for this simulation is available from `http://arxiv.org/src/1203.6127/anc`.

In the execution, we counted the number of iterations, (executions of Rebasing in Section 3.7.3), the sum of upper bounds on the number of multiplications and divisions given in Eqs. (15), (16), (17), (18), (19), (20), (23), (27) and (28), and the number of codewords found. Note also that Eq. (11) instead of Eq. (13) is used.

The parameter τ is set to the same as the number of generated errors in each simulation condition. N or R in the number of errors in Tables 2 and 3 indicates that the error vector is generated toward another codeword nearest from the transmitted codeword or completely randomly, respectively. The distribution of codewords is uniform on C_Γ. That of error vectors is uniform on the vectors of Hamming weight τ.

In the code construction, we always try to use the Feng-Rao improved construction. Specifically, for a given designed distance δ, we choose $\Gamma = \{s \in \widehat{H}(Q) \mid \lambda(s) = \nu(s) \geq \delta\}$, and construct C_Γ of Eq. (4). In the following, the designed distance is denoted by $d_{\mathrm{AG}}(C_\Gamma)$. It can be seen from Tables 1–4 and the following subsections that the computational complexity of the proposed algorithm tends to explode when the number of errors exceeds the error-correcting capability of the Guruswami-Sudan algorithm [24].

5.2 Comparison among the Three Proposed Termination Criteria

In Section 3.6 we proposed three criteria for terminating iteration of the proposed algorithm. From Tables 1–4, one can see the following. The first criterion has the smallest number of iterations, and the second is the second smallest. On the other hand, the first criterion has the largest number of multiplications and divisions. The second and the third have the similar numbers. Only the first criterion was proposed in [22] and we see that the new criteria are better than the old one.

The reason is as follows: The computation of quotient α_0/α_1 at Step 1 in Section 3.6.1 is costlier than updating $f_i^{(s)}$ and $g_i^{(s)}$ in Section 3.7.3 and the first criterion computes α_0/α_1 many times, which cancels the effect of decrease in the number of iterations. On the other hand, the second criterion computes α_0/α_1 only once, so it has the smaller number of multiplications and divisions than the first.

The second criterion is faster when $2\tau < d_{\mathrm{AG}}(C_\Gamma)$, while the third tends to be faster when $2\tau \geq d_{\mathrm{AG}}(C_\Gamma)$. In addition to this, the ratio of the number of iterations in the second criterion to that of the third is smaller with $2\tau < d_{\mathrm{AG}}(C_\Gamma)$ than with $2\tau \geq d_{\mathrm{AG}}(C_\Gamma)$. We speculate the reason behind them as follows: When $2\tau \geq d_{\mathrm{AG}}(C_\Gamma)$ and a wrong candidate is chosen at Eq. (26), after several iterations of Sections 3.6 and 3.7, we often observe in our simulation that no candidate satisfies Eq. (26) and the iteration stops automatically. Under such situation, the second criterion does not help much to decrease the number of iterations nor the computational complexity when a wrong candidate is chosen at Eq. (26), and there are many occasions at which a wrong candidate is chosen at Eq. (26) when $2\tau \geq d_{\mathrm{AG}}(C_\Gamma)$. On the other hand, when $2\tau < d_{\mathrm{AG}}(C_\Gamma)$, the second criterion helps to determine the transmitted information earlier than the third.

5.3 Tightness of Upper Bounds (31) and (32)

In Table 4, we observe that the upper bounds (31) and (32) are much larger than the actual number of iterations for $\tau = 5$. The disappearance of candidates satisfying Eq. (26) in the last paragraph may also explain the reason behind the large differences for $\tau = 5$.

On the other hand, we observe that the upper bound (32) is quite tight for $\tau = 5$ in Table 1 and $\tau = 10$N in Table 3. This suggests that improvement of Eq. (32) may need some additional assumption.

5.4 Klein Quartic, $(d_{\mathrm{AG}}(C_\Gamma), \tau) = (4, 1)$ or $(10, 4)$

We can use [4, 6, 12, 13, 32] to decode this set of parameters. It is essentially the forward elimination in the Gaussian elimination, and it takes roughly $n^3/3$ multiplications. In this case $n^3/3 = 4,055$. The proposed algorithm has the lower complexity than [4, 6, 12, 13, 32].

5.5 Klein Quartic, $(d_{\mathrm{AG}}(C_\Gamma), \tau) = (4, 2)$ or $(4, 3)$

The code is C_u with $u = 20$, $\dim C_u = 18$. There is no previously known algorithm that can handle this case.

5.6 Klein Quartic, $(d_{\mathrm{AG}}(C_\Gamma), \tau) = (10, 5)$

The code is C_u with $u = 13$, $\dim C_u = 11$. According to Beelen and Brander [5, Figure 1], we can use the original Guruswami-Sudan [24] but it seems that its faster variants cannot be used. We need the multiplicity 7 to correct 5 errors. We have to solve a system of $23(7 + 1)7/2 = 644$ linear equations. It takes $644^3/3 = 89,029,994$ multiplications in $\mathbf{F}_8$. The proposed algorithm is much faster.

5.7 Klein Quartic, $(d_{\mathrm{AG}}(C_\Gamma), \tau) = (10, 6)$

The code is C_u with $u = 13$, $\dim C_u = 11$. There is no previously known algorithm that can handle this case.

5.8 Hermitian, $(d_{\mathrm{AG}}(C_\Gamma), \tau) = (6, 2)$ or $(20, 9)$

We can use the BMS algorithm [45, 46] for this case. The complexity of [45, 46] is estimated as $O(a_1 n^2)$ and $a_1 n^2 = 24,576$. The complexity of the proposed algorithm seems comparable to [45, 46]. However, we are not sure which one is faster.

5.9 Hermitian, $(d_{\mathrm{AG}}(C_\Gamma), \tau) = (6, 3)$ or $(6, 4)$

The code becomes the Feng-Rao improved code with designed distance 6. Its dimension is 55. In order to have the same dimension by C_u we have to set $u = 60$, whose AG bound [3] is 4 and the Guruwsami-Sudan can correct up to 2 errors. The proposed algorithm finds all codewords in the improved code with 3 and 4 errors.

5.10 Hermitian, $(d_{\mathrm{AG}}(C_\Gamma), \tau) = (20, 10)$

The code is C_u with $u = 44$. The required multiplicity is 11, and the required designed list size is 14. The fastest algorithm for the interpolation step seems [5]. Beelen and Brander [5, Example 4] estimates the complexity of their algorithm as $O(\lambda^5 n^2 (\log \lambda n)^2 \log(\log \lambda n))$, where λ is the designed list size. Ignoring the log factor and assuming the scaling factor one in the big-O notation, the number of multiplications and divisions is $\lambda^5 n^2 = 2,202,927,104$. The proposed algorithm needs much fewer number of multiplications and divisions in $\mathbf{F}_{16}$.

5.11 Hermitian, $(d_{\mathrm{AG}}(C_\Gamma), \tau) = (20, 11)$

The Guruwsami-Sudan algorithm [24] can correct up to 10 errors and there seems no previously known algorithm that can handle this case.

5.12 Garcia-Stichtenoth (Example 2), $(d_{\mathrm{AG}}(C_\Gamma), \tau) = (6, 2)$, $(10, 4)$, or $(20, 9)$

We can use [4, 6, 12, 13, 32] to decode this set of parameters. It is essentially the forward elimination in the Gaussian elimination, and it takes roughly $n^3/3$ multiplications. In this case $n^3/3 = 152,177$. The proposed algorithm has the lower complexity than [4, 6, 12, 13, 32].

5.13 Garcia-Stichtenoth (Example 2), $(d_{\mathrm{AG}}(C_\Gamma), \tau) = (6, 3)$

This is a Feng-Rao improved code with dimension 58. In order to realize a code with the same dimension, we have to set $u = 79$ in C_u. The Guruswami-Sudan algorithm [24] can correct no error in this set of parameters. There seems no previously known algorithm that can handle this case.

5.14 Garcia-Stichtenoth (Example 2), $(d_{\mathrm{AG}}(C_\Gamma), \tau) = (10, 5)$

This is a Feng-Rao improved code with dimension 52. In order to realize a code with the same dimension, we have to set $u = 73$ in C_u. The Guruswami-Sudan algorithm [24] can correct 2 errors in this set of parameters. There seems no previously known algorithm that can handle this case.

5.15 Garcia-Stichtenoth (Example 2), $(d_{\mathrm{AG}}(C_\Gamma), \tau) = (20, 10)$

This is an ordinary one-point AG code C_u with $u = 58$ and dimension 37. The Guruswami-Sudan algorithm [24] can correct 10 errors with the multiplicity 154 and the designed list size 178. We have to solve a system of $77 \times (154 + 1)154/2 = 918,995$ linear equations. It takes $918,995^3/3 = 258,712,963,551,308,291$ multiplications in $\mathbf{F}_9$. The proposed algorithm is much faster.

5.16 Garcia-Stichtenoth (Example 2), $(d_{AG}(C_\Gamma), \tau) = (20, 11)$

The Guruswami-Sudan algorithm [24] can correct up to 10 errors and there seems no previously known algorithm that can handle this case.

6 Conclusion

In this paper, we modified the unique decoding algorithm for plane AG codes in [31] so that it can support one-point AG codes on *any* curve, and so that it can do the list decoding. The error correction capability of the original [31] and our modified algorithms are also expressed in terms of the minimum distance lower bound in [3].

We also proposed procedures to compute products and quotients in coordinate ring of affine algebraic curves, and by using those procedures we demonstrated that the modified decoding algorithm can be executed quickly. Specifically, its computational complexity is comparable to the BMS algorithm [45, 46] for one-point Hermitian codes, and much faster than the standard list decoding algorithms [5, 24] for several cases.

The original decoding algorithm [31] allows parallel implementation on circuits like the Kötter architecture [27]. Our modified algorithm retains this advantage. Moreover, if one can afford large circuit size, the proposed list decoding algorithm can be executed as quickly as the unique decoding algorithm by parallel implementation on a circuit.

Acknowledgment

The authors thank an anonymous reviewer for his/her careful reading that improved the presentation. This research was partially supported by the MEXT Grant-in-Aid for Scientific Research (A) No. 23246071, the Villum Foundation through their VELUX Visiting Professor Programme 2011–2012, the Danish National Research Foundation and the National Science Foundation of China (Grant No. 11061130539) for the Danish-Chinese Center for Applications of Algebraic Geometry in Coding Theory and Cryptography, the Spanish grant MTM2007-64704, and the Spanish MINECO grant No. MTM2012-36917-C03-03. The computer experiments in this research was conducted on Singular 3.1.3 [9].

References

[1] W. W. Adams and P. Loustaunau. *An Introduction to Gröbner Bases*, volume 3 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 1994.

[2] M. Ali and M. Kuijper. A parametric approach to list decoding of Reed-Solomon codes using interpolation. *IEEE Trans. Inform. Theory*, 57(10):6718–6728, Oct. 2011. doi:10.1109/TIT.2011.2165803. arXiv:1011.1040.

[3] H. E. Andersen and O. Geil. Evaluation codes from order domain theory. *Finite Fields Appl.*, 14(1):92–123, Jan. 2008. doi:10.1016/j.ffa.2006.12.004.

[4] P. Beelen. The order bound for general algebraic geometric codes. *Finite Fields Appl.*, 13(3):665–680, July 2007. doi:10.1016/j.ffa.2006.09.006.

[5] P. Beelen and K. Brander. Efficient list decoding of a class of algebraic-geometry codes. *Adv. Math. Commun.*, 4(4):485–518, 2010. doi:10.3934/amc.2010.4.485.

[6] P. Beelen and T. Høholdt. The decoding of algebraic geometry codes. In E. Martínez-Moro, C. Munuera, and D. Ruano, editors, *Advances in Algebraic Geometry Codes*, volume 5 of *Coding Theory and Cryptology*, pages 49–98. World Scientific, 2008. doi:10.1142/9789812794017_0002.

[7] M. Bras-Amorós and M. E. O'Sullivan. The correction capability of the Berlekamp-Massey-Sakata algorithm with majority voting. *Applicable Algebra in Engineering, Communication and Computing*, 17(5):315–335, Oct. 2006. doi:10.1007/s00200-006-0015-8.

[8] H. Chen. On the number of correctable errors of the Feng-Rao decoding algorithm for AG codes. *IEEE Trans. Inform. Theory*, 45(5):1709–1712, July 1999. doi:10.1109/18.771252.

[9] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann. SINGULAR 3-1-3 — A computer algebra system for polynomial computations, 2011. URL http://www.singular.uni-kl.de.

[10] I. M. Duursma. On erasure decoding of AG-codes. In *Proc. 1994 IEEE Information Theory Workshop*, Moscow, Russia, July 1994. URL http://www.math.uiuc.edu/~duursma/pub/Erasure94.pdf.

[11] I. M. Duursma. Coset bound / shift bound. private communication, Nov. 2012.

[12] I. M. Duursma and S. Park. Coset bounds for algebraic geometric codes. *Finite Fields Appl.*, 16(1):36–55, Jan. 2010. doi:10.1016/j.ffa.2009.11.006.

[13] I. M. Duursma, R. Kirov, and S. Park. Distance bounds for algebraic geometric codes. *J. Pure Appl. Algebra*, 215(8):1863–1878, Aug. 2011. doi:10.1016/j.jpaa.2010.10.018. arXiv:1001.1374.

[14] D. Eisenbud. *Commutative Algebra with a View Toward Algebraic Geometry*, volume 150 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 1995.

[15] H. Elbrønd Jensen, R. R. Nielsen, and T. Høholdt. Performance analysis of a

decoding algorithm for algebraic-geometry codes. *IEEE Trans. Inform. Theory*, 45(5):1712–1717, July 1999. doi:10.1109/18.771253.

[16] G. L. Feng and T. R. N. Rao. Decoding algebraic geometric codes up to the designed minimum distance. *IEEE Trans. Inform. Theory*, 39(1):36–47, Jan. 1993. doi:10.1109/18.179340.

[17] G. L. Feng and T. R. N. Rao. Improved geometric Goppa codes part I, basic theory. *IEEE Trans. Inform. Theory*, 41(6):1678–1693, Nov. 1995. doi:10.1109/18.476241.

[18] M. Fujisawa, H. Matsui, M. Kurihara, and S. Sakata. With a higher probability one can correct errors up to half the designed distance for primal codes from curves. In *Proc. SITA2006*, pages 101–104, Hakodate, Hokkaido, Japan, Nov. 2006.

[19] A. Garcia and H. Stichtenoth. A tower of Artin-Schreier extensions of function fields, attaining the Drinfeld-Vladut bound. *Invent. Math.*, 121(1):211–222, July 1995. doi:10.1007/BF01884295.

[20] O. Geil and R. Pellikaan. On the structure of order domains. *Finite Fields Appl.*, 8(3):369–396, July 2002. doi:10.1006/ffta.2001.0347.

[21] O. Geil, C. Munuera, D. Ruano, and F. Torres. On the order bounds for one-point AG codes. *Adv. Math. Commun.*, 5(3):489–504, 2011. doi:10.3934/amc.2011.5.489. arXiv:1002.4759.

[22] O. Geil, R. Matsumoto, and D. Ruano. List decoding algorithms based on Gröbner bases for general one-point AG codes. In *Proc. ISIT 2012*, pages 86–90, Cambridge, MA, USA, July 2012. doi:10.1109/ISIT.2012.6284685. arXiv:1201.6248.

[23] O. Geil, R. Matsumoto, and D. Ruano. Feng-Rao decoding of primary codes. *Finite Fields Appl.*, 23:35–52, Sept. 2013. doi:10.1016/j.ffa.2013.03.005. arXiv:1210.6722.

[24] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Trans. Inform. Theory*, 45(4):1757–1767, Sept. 1999. doi:10.1109/18.782097.

[25] T. Høholdt and R. Pellikaan. On the decoding of algebraic-geometric codes. *IEEE Trans. Inform. Theory*, 41(6):1589–1614, Nov. 1995. doi:10.1109/18.476214.

[26] J. Justesen and T. Høholdt. *A Course in Error-Correcting Codes*. EMS Textbooks in Mathematics. European Mathematical Society Publishing House, Zürich, Switzerland, 2004.

[27] R. Kötter. A fast parallel implementation of a Berlekamp-Massey algorithm for algebraic-goemetric codes. *IEEE Trans. Inform. Theory*, 44(4):1353–1368, July 1998. doi:10.1109/18.681314.

[28] R. F. Lax. Generic interpolation polynomial for list decoding. *Finite Fields Appl.*, 18(1):167–178, Jan. 2012. doi:10.1016/j.ffa.2011.07.007.

[29] K. Lee and M. E. O'Sullivan. List decoding of Reed-Solomon codes from a Gröbner basis perspective. *J. Symbolic Comput.*, 43(9):645–658, Sept. 2008.

doi:10.1016/j.jsc.2008.01.002.

[30] K. Lee and M. E. O'Sullivan. List decoding of Hermitian codes using Gröbner bases. *J. Symbolic Comput.*, 44(12):1662–1675, Dec. 2009. doi:10.1016/j.jsc.2007.12.004. arXiv:cs/0610132.

[31] K. Lee, M. Bras-Amorós, and M. E. O'Sullivan. Unique decoding of plane AG codes via interpolation. *IEEE Trans. Inform. Theory*, 58(6):3941–3950, June 2012. doi:10.1109/TIT.2012.2182757. arXiv:1110.6251.

[32] R. Matsumoto and S. Miura. On the Feng-Rao bound for the $\mathcal{L}$-construction of algebraic geometry codes. *IEICE Trans. Fundamentals*, E83-A(5):926–930, May 2000. URL http://www.rmatsumoto.org/repository/e83-a_5_923.pdf.

[33] R. Matsumoto and S. Miura. Finding a basis of a linear system with pairwise distinct discrete valuations on an algebraic curve. *J. Symbolic Comput.*, 30(3):309–323, Sept. 2000. doi:10.1006/jsco.2000.0372.

[34] R. Matsumoto and S. Miura. On construction and generalization of algebraic geometry codes. In T. Katsura et al., editors, *Proc. Algebraic Geometry, Number Theory, Coding Theory, and Cryptography*, pages 3–15, Univ. Tokyo, Japan, Jan. 2000. URL http://www.rmatsumoto.org/repository/weight-construct.pdf.

[35] R. Matsumoto, D. Ruano, and O. Geil. Generalization of the Lee-O'Sullivan list decoding for one-point AG code. *J. Symbolic Comput.*, 55:1–9, Aug. 2013. doi:10.1016/j.jsc.2013.03.001. arXiv:1203.6129.

[36] C. P. McKeague. *Elementary Algebra*. Brooks Cole, Florence, KY 41022-6904, USA, 9th edition, 2012. ISBN 0840064217.

[37] S. Miura. Algebraic geometric codes on certain plane curves. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 76(12):1–13, Dec. 1993. doi:10.1002/ecjc.4430761201. (original Japanese version published as Trans. IEICE, vol. J75-A, no. 11, pp. 1735–1745, Nov. 1992).

[38] S. Miura. Linear codes on affine algebraic curves. *Trans. IEICE*, J81-A(10):1398–1421, Oct. 1998.

[39] R. Pellikaan. On the efficient decoding of algebraic-geometric codes. In P. Camion, P. Charpin, and S. Harari, editors, *Eurocode '92 International Symposium on Coding Theory and Applications*, number 339 in CISM Courses and Lectures, pages 231–253. CISM International Centre for Mechanical Sciences, Springer, 1993. URL http://www.win.tue.nl/~ruudp/paper/17.pdf.

[40] J. C. Rosales and P. A. García-Sánchez. *Numerical Semigroups*, volume 20 of *Developments in Mathematics*. Springer, New York, 2009. ISBN 978-1-4419-0159-0.

[41] K. Saints and C. Heegard. Algebraic-geometric codes and multidimensional cyclic codes: A unified theory and algorithms for decoding using Gröbner bases. *IEEE Trans. Inform. Theory*, 41(6):1733–1751, Nov. 1995. doi:10.1109/18.476246.

[42] S. Sakata. On fast interpolation method for Guruswami-Sudan list decoding of one-point algebraic-geometry codes. In S. Boztaş and I. E. Shparlin-

ski, editors, *Proc. AAECC-14*, volume 2227 of *Lecture Notes in Computer Science*, pages 172–181, Melbourne, Australia, Nov. 2001. Springer-Verlag. doi:10.1007/3-540-45624-4_18.

[43] S. Sakata. Multivariate interpolation and list decoding. In K. Kobayashi and H. Morita, editors, *Proc. 3rd Asian-European Workshop on Information Theory*, pages 28–31, Kamogawa, Chiba, Japan, June 2003. Society of Information Theory and its Applications. ISBN 4-902087-04-9.

[44] S. Sakata and M. Fujisawa. Fast decoding of multipoint codes from algebraic curves up to the order bound. In *Proc. SITA2011*, pages 417–422, Iwate, Japan, Nov. 2011.

[45] S. Sakata, H. Elbrønd Jensen, and T. Høholdt. Generalized Berlekamp-Massey decoding of algebraic-geometric codes up to half the Feng-Rao bound. *IEEE Trans. Inform. Theory*, 41(5):1762–11768, Sept. 1995. doi:10.1109/18.476248.

[46] S. Sakata, J. Justesen, Y. Madelung, H. Elbrønd Jensen, and T. Høholdt. Fast decoding of algebraic geometric codes up to the designed minimum distance. *IEEE Trans. Inform. Theory*, 41(5):1672–1677, Sept. 1995. doi:10.1109/18.476240.

[47] J. Schicho. Inversion of birational maps with Gröbner bases. In B. Buchberger and F. Winkler, editors, *Gröbner Bases and Applications*, volume 251 of *London Mathematical Society Lecture Note Series*, pages 495–503. Cambridge University Press, 1998. ISBN 9780521632980. doi:10.1017/CBO9780511565847.031.

[48] M. A. Shokrollahi and H. Wasserman. List decoding of algebraic-geometric codes. *IEEE Trans. Inform. Theory*, 45(2):432–437, Mar. 1999. doi:10.1109/18.748993.

[49] L.-Z. Tang. A Gröbner basis criterion for birational equivalence of affine varieties. *J. Pure Appl. Algebra*, 123(1-3):275–283, Jan. 1998. doi:10.1016/S0022-4049(97)00139-4.

[50] D. Umehara and T. Uyematsu. One-point algebraic geometric codes from Artin-Schreier extensions of Hermitian function fields. *IEICE Trans. Fundamentals*, E81-A(10):2025–2031, Oct. 1998.

[51] W. V. Vasconcelos. *Computational Methods in Commutative Algebra and Algebraic Geometry*, volume 2 of *Algorithms and Computation in Mathematics*. Springer-Verlag, Berlin, 1998.

[52] C. Voss and T. Høholdt. An explicit construction of a sequence of codes attaining the Tsfasman-Vlăduţ-Zink bound. *IEEE Trans. Inform. Theory*, 43(1):128–135, Jan. 1997. doi:10.1109/18.567659.