Part 3) of our Main Theorem is obtained from the next result.

Proposition 13: If $q^r + 1 \leq m \leq q^{2r+1} - q^{r+1} - q + 1$, then the mapping

$$\psi: \Gamma \to \mathrm{Aut}(C_m)$$

is an isomorphism of groups.

Proof: It is easy to prove that ψ is a homomorphism of groups. Now suppose that $\psi(\sigma) = 1$, then $\pi_\sigma = 1$. Hence, the equality $\sigma(P_{a,b}) = P_{a,b}$ holds for all places $P_{a,b}$. This implies that $\sigma^{-1}(x) = x$ and $\sigma^{-1}(y) = y$, hence $\sigma = 1$. Thus, ψ is injective.

We show that ψ is surjective. Let $\pi \in \mathrm{Aut}(C_m)$. From Proposition 10, we have an automorphism $\sigma \in \Gamma$ such that

$$\pi(ev_D(x)) = ev_D(\sigma(x)) \quad \text{and} \quad \pi(ev_D(y)) = ev_D(\sigma(y)).$$

This means that

$$\pi(a, b) = (\sigma(x)(P_{a,b}), \sigma(y)(P_{a,b})).$$

Therefore, we have $\pi = \pi_{\sigma^{-1}}$, that is, $\pi = \psi(\sigma^{-1})$. □

ACKNOWLEDGMENT

The authors wish to thank one of the referees for valuable comments and careful reviews on the first version of this correspondence.

REFERENCES

[1] A. Garcia, "On Goppa codes and Artin–Schreier extensions," *Comm. Algebra*, vol. 20, pp. 3683–3689, 1992.

[2] S. Kondo and H. Momiyama, "Automorphism group of Hermitian code C_4 over $GF(4)$," *Gakujutsu Kenkyu*, vol. 49, pp. 17–23, 2001. School of Education, Waseda University, Tokyo, Japan.

[3] J. H. van Lint and T. A. Springer, "Generalized Reed–Solomon codes from algebraic geometry," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 305–309, May 1987.

[4] D. J. S. Robinson, *A Course in the Theory of Groups*. Berlin Heidelberg: Springer-Verlag, 1995.

[5] H. Stichtenoth, "Uber die Automorphismengruppe eines algebraischen Funktionenkörpers von Primzahlcharakteristik, Tail II," *Arch. Math.*, vol. 24, pp. 615–631, 1973.

[6] ——, "A note on Hermitian codes over $\mathrm{GF}(q^2)$," *IEEE Trans. Inform. Theory*, vol. 34, pp. 1345–1348, Sept. 1988.

[7] ——, "On automorphisms of geometric Goppa codes," *J. Algebra*, vol. 130, pp. 113–121, 1990.

[8] ——, *Algebraic Function Fields and Codes*. Berlin Heidelberg: Springer-Verlag, 1993, Universitext.

[9] H. J. Tiersma, "Remarks on codes from Hermitian curves," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 605–609, July 1987.

[10] S. Wesemeyer, "On the automorphism group of various Goppa codes," *IEEE Trans. Inform. Theory*, vol. 44, pp. 630–643, Mar. 1998.

[11] C. Xing, "On automorphism groups of the Hermitian codes," *IEEE Trans. Inform. Theory*, vol. 41, pp. 1629–1635, Nov. 1995.

Efficient Root-Finding Algorithm With Application to List Decoding of Algebraic–Geometric Codes

Xin-Wen Wu, *Member, IEEE,* and Paul H. Siegel, *Fellow, IEEE*

***Abstract*—A list decoding for an error-correcting code is a decoding algorithm that generates a list of codewords within a Hamming distance t from the received vector, where t can be greater than the error-correction bound. In [18], a list-decoding procedure for Reed–Solomon codes [19] was generalized to algebraic–geometric codes. A recent work [8] gives improved list decodings for Reed–Solomon codes and algebraic-geometric codes that work for all rates and have many applications. However, these list-decoding algorithms are rather complicated. In [17], Roth and Ruckenstein proposed an efficient implementation of the list decoding of Reed–Solomon codes. In this correspondence, extending Roth and Ruckenstein's fast algorithm for finding roots of univariate polynomials over polynomial rings, i.e., the Reconstruct Algorithm, we will present an efficient algorithm for finding the roots of univariate polynomials over function fields. Based on the extended algorithm, we give an efficient list-decoding algorithm for algebraic-geometric codes.**

***Index Terms*—Algebraic-geometric codes, list decoding, root-finding algorithm.**

I. INTRODUCTION

Suppose C is an $[n, k, d]$ code over the finite field $\boldsymbol{F}_q$, $t < n$ is a positive integer. For any received vector $\boldsymbol{y} = (y_1, \ldots, y_n) \in \boldsymbol{F}_q^n$, we refer to any codeword $\boldsymbol{c}$ in C satisfying $d(\boldsymbol{c}, \boldsymbol{y}) \leq t$ as a t*-consistent codeword*. A decoding problem is, in fact, the problem of finding an effective (or efficient) algorithm which can find t-consistent codewords, and we call such an algorithm a decoding algorithm that can correct t errors. The *classical decodings* (sometimes called *unique decodings*) of error-correcting codes consider the algorithms which can correct $\tau = \lfloor \frac{d-1}{2} \rfloor$ or fewer errors [5], [11]. It is clear that in any Hamming sphere in $\boldsymbol{F}_q^n$ of radius $\leq \tau$, there exists at most one codeword of an $[n, k, d]$ code. We call τ the *error-correction bound* of the code. On the other hand, if the number of errors $t > \tau$, then there may exist several distinct t-consistent codewords. A *list decoding* is a decoding algorithm which tries to construct a list of t-consistent codewords. Thus, a list-decoding algorithm makes it possible to recover the information from errors beyond the traditional error-correction bound.

List decoding was introduced by Elias [4] and Wozencraft [20]. In [19], Sudan proposed a list-decoding algorithm for Reed–Solomon codes. Shokrollahi and Wasserman generalized Sudan's algorithm and derived a list-decoding scheme for algebraic-geometric codes [18]. These algorithms are effective only for codes of relatively low rates. In a recent paper [8], Guruswami and Sudan proposed improved algorithms for Reed–Solomon and algebraic-geometric codes. The algorithms have better error-correction capabilities than previous algorithms for any code rate. However, the implementations

Manuscript received April 12, 2000; revised April 24, 2001. This work was supported by the National Science Foundation under Grant NCR-9612802 and by the National Storage Industry Consortium. The material in this correspondence was presented in part at the IEEE International Symposium on Information Theory, Sorrento, Italy, June 25–30, 2000.

X.-W. Wu was with the Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093-0407 USA. He is now with the Academy of Mathematics and System Sciences, Chinese Academy of Sciences, Beijing, 100080, China (e-mail: wxw@math08.math.ac.cn).

P. H. Siegel is with the Center for Magnetic Recording Research, University of California, San Diego, La Jolla, CA 92093-0401 USA (e-mail: psiegel@ucsd.edu).

Communicated by I. F. Blake, Associate Editor for Coding Theory.
Publisher Item Identifier S 0018-9448(01)06482-3.

of the list-decoding algorithms are rather complicated, especially for algebraic-geometric codes. As we will discuss in the next section, the list-decoding procedures consist of two main steps. The first step is, in fact, reduced to the problem of solving a system of homogeneous linear equations, which can be implemented with low complexity using Gaussian elimination. The second step is a problem of finding roots in some spaces of univariate polynomials $H(T)$ over polynomial rings for Reed–Solomon codes and over function fields for algebraic-geometric codes, respectively.

Shokrollahi and Wasserman [18] and Guruswami and Sudan [8] proposed factorization (or root-finding) algorithms to find the roots of $H(T)$, but the implementation of these algorithms is rather arduous. Gao and Shokrollahi [6] designed an algorithm for computing roots of polynomials over the fields of rational functions on plane curves. Their work also includes an algorithm for finding roots of $H(T)$ over polynomial rings. In [9], Høholdt and Nielsen studied fast list decoding for Hermitian codes. They transformed the factorization of $H(T)$ over the Hermitian function field into a problem of factoring a univariate polynomial over a large finite field. Their algorithm remains to be extended to general algebraic-geometric codes. In [1], Augot and Pecquet proposed root-finding algorithms. Augot and Pecquet's algorithms do not work for the improved list decodings for Reed–Solomon and algebraic-geometric codes in [8]. Recently, Roth and Ruckenstein [17] presented a fast list-decoding scheme for Reed–Solomon codes. They sped up the first step of the list-decoding algorithm for Reed–Solomon codes in [19], making use of special properties of the system of homogeneous linear equations that arises. More importantly, based upon a different approach, they proposed an efficient algorithm for finding roots of univariate polynomials over polynomial rings to accelerate the second step of the list decoding for Reed–Solomon codes.

In this correspondence, extending Roth and Ruckenstein's algorithm, we derive an efficient root-finding algorithm for finding roots of polynomials over function fields. As an application of the root-finding algorithm we then present an efficient list-decoding procedure for algebraic-geometric codes.

In the next section, we will give the basic definitions and properties of algebraic-geometric codes and a statement of the root-finding problem. In Section III, we will present an efficient root-finding algorithm for finding roots of polynomials over function fields and prove the correctness of the algorithm. The efficient list-decoding algorithm for algebraic-geometric codes will be given in Section IV. In Section V, we present our conclusions. We will give an example and the complexity analysis for the root-finding algorithm in Appendixes A and B, respectively.

II. Preliminaries

Let $\mathcal{X}$ be a nonsingular, absolutely irreducible curve in $\boldsymbol{P}_k^m$, the m-dimensional projective space over a field k. Denote by $k(\mathcal{X})$ the function field of $\mathcal{X}$ over k. We can view $\mathcal{X}$ as a curve over $\overline{k}$, where $\overline{k}$ is the algebraic closure of k. Over $\overline{k}$ there is a one-to-one correspondence between the points P of $\mathcal{X}$ and the discrete valuation rings of the function field. When k is not algebraically closed, we cannot see all points of $\mathcal{X}$ over k; nevertheless, we can look at the discrete valuation rings contained in $k(\mathcal{X})$ such that the discrete valuation is trivial on k. Let v be a discrete valuation of $k(\mathcal{X})$ and R_v be its valuation ring with maximal ideal m_v, we call the pair (R_v, m_v) a closed point of $\mathcal{X}$. The degree $\deg(P)$ of P is defined as $[R_v/m_v : k]$ where R_v is the corresponding discrete valuation ring, which is a positive integer since the field R_v/m_v is a finite extension of k. A point P of $\mathcal{X}$ with $\deg(P) = 1$ is called a rational point.

A divisor of $\mathcal{X}$ is a formal linear combination

$$D = \sum n_P P$$

where the sum is over all closed points of $\mathcal{X}$, n_P are integers, and all but finitely many n_P's are zero. The degree of D is

$$\deg(D) = \sum n_P \cdot \deg(P).$$

The support of D is $\sup(D) = \{P \mid n_P \neq 0\}$.

Let P be a closed point of $\mathcal{X}$. In the sequel, we denote by ord_P the discrete valuation associated to P. We recall that for any nonzero rational function $\varphi \in k(\mathcal{X})$ there are only finitely many closed points P such that $\mathrm{ord}_P(\varphi) \neq 0$. If $\mathrm{ord}_P(\varphi) > 0$, φ is said to have a zero of order $\mathrm{ord}_P(\varphi)$ at P, if $\mathrm{ord}_P(\varphi) < 0$, φ is said to have a pole of order $-\mathrm{ord}_P(\varphi)$ at P. For a nonzero rational function $\varphi \in k(\mathcal{X})$, the divisor of φ is defined as

$$(\varphi) = \sum \mathrm{ord}_P(\varphi) P,$$

where the sum is over all closed points. Let

$$(\varphi)_0 = \sum_{\mathrm{ord}_P(\varphi) > 0} \mathrm{ord}_P(\varphi) P$$

and

$$(\varphi)_\infty = \sum_{\mathrm{ord}_P(\varphi) < 0} -\mathrm{ord}_P(\varphi) P.$$

Then, $(\varphi) = (\varphi)_0 - (\varphi)_\infty$. It can be shown that for any nonzero rational function φ, the degree of (φ) is zero, i.e., $\deg(\varphi) = 0$.

If we define

$$D + D' = \sum (n_P + n'_P) P$$

where $D = \sum n_P P$ and $D' = \sum n'_P P$ are any two divisors of $\mathcal{X}$, then the set of divisors of $\mathcal{X}$ forms an additive group $\mathrm{Div}(\mathcal{X})$. A divisor $D = \sum n_P P$ is called effective and denoted as $D \geq 0$ if all n_P are nonnegative. If

$$D - D' = \sum (n_P - n'_P) P \geq 0$$

we denote $D \geq D'$. Define

$$L(D) = \{f \mid f \in k(\mathcal{X}),\ f = 0 \text{ or } (f) + D \geq 0\}.$$

It can be proved that $L(D)$ is a linear space over k.

Now suppose $\mathcal{X}$ is a nonsingular, absolutely irreducible curve in the m-dimensional projective space $\boldsymbol{PF}_q^m$ over the finite field $\boldsymbol{F}_q$. Suppose $\{P_1, P_2, \ldots, P_n\}$ is a set of rational points of $\mathcal{X}$. Let $D = P_1 + \cdots + P_n$, and G be another divisor of $\mathcal{X}$ satisfying $\sup(D) \cap \sup(G) = \emptyset$. An *algebraic-geometric code* (or AG code, for short) $C_L(D, G)$ is defined as

$$C_L(D, G) = \{(f(P_1), f(P_2), \ldots, f(P_n)) \mid f \in L(G)\}.$$

Suppose $\rho = \deg G < n$; then, $C_L(D, G)$ has length n, dimension $\geq \rho - g + 1$, and minimum distance $\geq n - \rho$, where g is the genus of the curve.

In this correspondence, we consider the AG codes $C_L(D, G)$ with $D = P_1 + \cdots + P_n$ and $G = \rho P$, such that ρ is an integer and $\{P_1, \ldots, P_n, P\}$ is the set of all the rational points of $\mathcal{X}$. These codes include Hermitian codes as special cases and are of special interest in practical applications.

By the definitions, we know that if $D \geq D'$ then $L(D) \supseteq L(D')$. Let ρ be a nonnegative integer and P be a point of $\mathcal{X}$. If $L(\rho P) \neq L((\rho - 1)P)$, or equivalently there exists a rational function φ, such that φ has a pole only at P and the order of the pole of φ at P is ρ, i.e., $\mathrm{ord}_P(\varphi) = -\rho$, then we call ρ a *nongap* of P. Let $\{\rho_1, \rho_2, \rho_3, \ldots\}$ be the set of all the nongaps of P and $\rho_1 < \rho_2 < \rho_3 < \cdots$, and let g be the genus of the curve. Then

$$0 = \rho_1 < \rho_2 < \cdots < \rho_g < \rho_{g+1} = 2g,$$

and $\rho_i = i + g - 1$ when $i \geq g + 1$ (see [3]). Let $\varphi_1, \varphi_2, \varphi_3, \ldots$ be a sequence of rational functions, such that φ_i has a pole only at P and $\mathrm{ord}_P(\varphi_i) = -\rho_i$. Then it is easy to check that $\{\varphi_1, \varphi_2, \ldots, \varphi_i\}$ is a basis of $L(\rho_i P)$.

In the sequel, we call the list-decoding algorithm for AG codes proposed by Guruswami and Sudan in [8] the G–S Algorithm. Given a received vector $\boldsymbol{y} = (y_1, \ldots, y_n)$, the G–S Algorithm first finds a nontrivial polynomial $H(T)$ with coefficients in the function field of $\mathcal{X}$ over $\boldsymbol{F}_q$ satisfying some conditions. It can be proved that a polynomial satisfying such conditions does exist. Also, in [8] the authors proved that if $f \in L(\rho P)$ is such that $f(P_i) = y_i$ for at least e values of $i \in \{1, 2, \ldots, n\}$, then $H(f) = 0$, i.e., $H(f)$ is identically zero as a rational function. This means that if $(f(P_1), \ldots, f(P_n)) \in C_L(D, \rho P)$ is a t-consistent codeword, where $t = n - e$, then $H(f) = 0$. So the problem of finding all the t-consistent codewords is reduced to the problem of finding all the roots in $L(\rho P)$ of $H(T)$.

For the precise statement of the G–S algorithm, please see [8]. Step 1, i.e., the step of finding a polynomial $H(T)$, can be reduced to a problem of solving a system of homogeneous linear equations over $\boldsymbol{F}_q$, where the unknowns are the coefficients of $H(T)$. This can be done by Gaussian elimination with low complexity. So, the complexity is mainly based on Step 2, i.e., the step of finding the roots in $L(\rho P)$ of $H(T)$. The purpose of this work is to find an efficient root-finding algorithm for finding roots of polynomials over function fields, and then give an efficient list-decoding algorithm, replacing Step 2 by the new root-finding algorithm. Our problem can be stated as follows.

The Root-Finding Problem: Let $\mathcal{X}$ be a nonsingular, absolutely irreducible curve defined over $\boldsymbol{F}_q$. Let P be a point of $\mathcal{X}$, which can be the point at infinity. Assume $L(\rho P)$ is a k-dimensional space, and $\varphi_1, \ldots, \varphi_k$ form a basis of $L(\rho P)$, where every φ_i has a pole only at P and $\mathrm{ord}_P(\varphi_i) = -\rho_i$. Given a nonzero polynomial

$$H(T) = h_0 + h_1 T + \cdots + h_s T^s$$

with $h_j \in L(lP)$ for $j = 0, 1, \ldots, s$, where l is a nongap of P, we want to find the roots in $L(\rho P)$ of $H(T)$.

III. Efficient Root-Finding Algorithm

In this section, we will derive an efficient algorithm for solving the root-finding problem. We first give a simple example to illustrate the idea.

Example 3.1: The Reed–Solomon codes can be viewed as special AG codes defined from a projective line. The projective line over $\boldsymbol{F}_4$ contains a point $P = [y : x] = [0 : 1]$, the point at infinity. The function $X = x/y$ is a rational function. We now try to find the roots in the space $\langle 1, X, X^2 \rangle$ of the following polynomial:

$$H(T) = T^2 + \left(X^2 + X + 1\right) T + \left(X^3 + X\right).$$

Suppose the roots have the form $f = f_0 + f_1 X + f_2 X^2$. To find the roots, we need to determine $f_0, f_1, f_2 \in \boldsymbol{F}_4$, such that

$$H\left(f_0 + f_1 X + f_2 X^2\right) = 0. \tag{3.1}$$

We introduce a method for determining f_2, f_1, and f_0 recursively. From (3.1), we have that the leading coefficient of $H(f_0 + f_1 X + f_2 X^2)$, which is equal to the leading coefficient of $H(f_2 X^2)$, is zero. By simple calculation, the leading coefficient of $H(f_2 X^2)$ is $f_2^2 + f_2$. So

$$f_2^2 + f_2 = 0.$$

From the equation we have $f_2 = 0$ or $f_2 = 1$.

For any root α_1 of $f_2^2 + f_2 = 0$. Set $H_2(T) = H_1(T + \alpha_1 X^2)$, where $H_1(T) := H(T)$. By (3.1), we have

$$H_2(f_0 + f_1 X) = 0.$$

Thus, the leading coefficient of $H_2(f_1 X)$ is zero. For $\alpha_1 = 0$, we get a polynomial equation $f_1 + 1 = 0$. We then find $f_1 = 1$. For another root $\alpha_1 = 1$, we get $f_1 = 0$.

Next, for any value α_2 of f_1, set $H_3(T) = H_2(T + \alpha_2 X)$. We have that $H_3(f_0) = 0$. So, the leading coefficient of $H_3(f_0)$ is zero. For $\alpha_2 = 1$, we find $f_0 = 0$. For $\alpha_2 = 0$, we get $f_0 = 1$.

Therefore, we find two roots $f = f_0 + f_1 X + f_2 X^2$ in $\langle 1, X, X^2 \rangle$ of $H(T)$, namely, $f = X$ and $f = X^2 + 1$. □

Roth and Ruckenstein in [17] presented an efficient algorithm for finding the roots of univariate polynomials over polynomial rings, i.e., the Reconstruct Algorithm. It is easy to see that the root-finding procedure in this example is equivalent to Roth and Ruckenstein's algorithm. In fact, one can easily verify that the equations for finding the coefficients f_i of any root of $H(T)$ are equal to the corresponding equations using Roth and Ruckenstein's algorithm.

Now consider the general case. Let

$$f = f_1 \varphi_1 + \cdots + f_k \varphi_k \in L(\rho P)$$

be a root in $L(\rho P)$ of polynomial $H(T)$. As in the example above, we can view $\varphi_1, \varphi_2, \ldots$ as formal variables and $H(f)$ as a polynomial in $\varphi_1, \varphi_2, \ldots$. The rational function φ_i has a pole of order ρ_i at P, and $\varphi_1^{i_1} \cdots \varphi_u^{i_u}$ has a pole of order $i_1 \rho_1 + \cdots + i_u \rho_u$ at P. We define the *weighted degree* of the monomial $\varphi_1^{i_1} \cdots \varphi_u^{i_u}$ as $i_1 \rho_1 + \cdots + i_u \rho_u$. Let $<_{\mathrm{WGL}}$ be the weighted graded lexicographic order. Under the order $<_{\mathrm{WGL}}$, and being reduced modulo the curve, the polynomial $H(f_1 \varphi_1 + \cdots + f_k \varphi_k)$ can be written uniquely as

$$\sum_{i_1, i_2, \ldots, i_u} a_{i_1, i_2, \ldots, i_u} \varphi_1^{i_1} \varphi_2^{i_2} \cdots \varphi_u^{i_u}$$

where $a_{i_1, i_2, \ldots, i_u}$ are elements of $\boldsymbol{F}_q$. We call any

$$a_{i_1, i_2, \ldots, i_u} \varphi_1^{i_1} \varphi_2^{i_2} \cdots \varphi_u^{i_u} \text{ with } a_{i_1, i_2, \ldots, i_u} \neq 0$$

a term of $H(f_1 \varphi_1 + \cdots + f_k \varphi_k)$ and $a_{i_1, i_2, \ldots, i_u}$ the coefficient of this term. The term with the greatest weighted degree is called the leading term and its coefficient the leading coefficient. It is clear that the leading term has the smallest discrete valuation ord_P, i.e., $a_{j_1, j_2, \ldots, j_u} \varphi_1^{j_1} \varphi_2^{j_2} \cdots \varphi_u^{j_u}$ is the leading term of $H(f)$ if and only if

$$\begin{aligned} &\mathrm{ord}_P\left(\varphi_1^{j_1} \varphi_2^{j_2} \cdots \varphi_u^{j_u}\right) \\ &\quad = \min \Big\{ \mathrm{ord}_P\left(\varphi_1^{i_1} \varphi_2^{i_2} \cdots \varphi_u^{i_u}\right) \Big| a_{i_1, i_2, \ldots, i_u} \varphi_1^{i_1} \varphi_2^{i_2} \cdots \varphi_u^{i_u} \\ &\qquad\qquad \text{is a term of } H(f) \Big\}. \end{aligned}$$

From $H(f_1 \varphi_1 + \cdots + f_k \varphi_k) = 0$, we have that the coefficient of every term must be zero. Thus, we get a system of polynomial equations over $\boldsymbol{F}_q$ with unknowns $f_1, \ldots, f_k$

$$\begin{cases} a_1(f_1, \ldots, f_k) = 0 \\ \quad\vdots \\ a_v(f_1, \ldots, f_k) = 0. \end{cases}$$

So, the roots in $L(\rho P)$ of $H(T)$ are given by finding the points over $\boldsymbol{F}_q$ of the affine variety in $\boldsymbol{F}_q[f_1, \ldots, f_k]$ defined by the system of equations above. Using the idea in Example 3.1, we present a recursive root-finding procedure in the following.

Root-Finding Procedure: For convenience, we denote $H_1(T) := H(T)$. From

$$H_1(f_1 \varphi_1 + \cdots + f_k \varphi_k) = 0 \tag{3.2}$$

we have that the leading coefficient of $H_1(f_1\varphi_1+\cdots+f_k\varphi_k)$, which is equal to the leading coefficient of $H_1(f_k\varphi_k)$, is zero. This is a polynomial equation over $\boldsymbol{F}_q$ with unknown f_k, we denote it by

$$g_1(f_k)=0.$$

Solving this equation, we can find f_k.

Suppose we have obtained $f_k, \ldots, f_{k-i+1}$. We, therefore, will have found a polynomial g_i over $\boldsymbol{F}_q$ such that f_{k-i+1} is a root of g_i. For each of the distinct roots α_i of the polynomial equation

$$g_i(f_{k-i+1})=0$$

set $H_{i+1}(T)=H_i(T+\alpha_i\varphi_{k-i+1})$. From the fact that the leading coefficient of $H_{i+1}(f_1\varphi_1+\cdots+f_{k-i}\varphi_{k-i})$ is zero, we get a polynomial equation over $\boldsymbol{F}_q$ with unknown f_{k-i}

$$g_{i+1}(f_{k-i})=0. \tag{3.3}$$

We then get f_{k-i} by solving the equation. Therefore, we obtain the coefficients $f_k, f_{k-1}, \ldots, f_1$ of any root f in $L(\rho P)$ of $H(T)$ recursively. □

With respect to the complexity of the root-finding procedure above, we note that there are efficient algorithms for finding the roots of polynomials over a finite field [2]. Another factor that affects the complexity is the size of the output set. Given a nontrivial polynomial $H(T)$, the root-finding procedure will output a set of sequences $[f_k, f_{k-1}, \ldots, f_1]$ corresponding to rational functions $f_1\varphi_1+\cdots+f_k\varphi_k$ in $L(\rho P)$. It can be proved that if the T-degree of $H(T)$ is s, then every g_i is a polynomial of degree less than or equal to s, which has at most s roots. The root-finding procedure above suggests that the number of root extractions grows exponentially. Although the polynomial $H(T)$ has at most $s=\deg(H(T))$ roots in $L(\rho P)$, we cannot prove that the size of the output set is bounded by s without sufficient information about the polynomials g_i. On the other hand, to prove the correctness of the root-finding procedure, we need to show that the polynomials g_i are nontrivial so that we can get f_{k-i+1} by solving the equation $g_i=0$. However, the root-finding procedure above does not give any explicit description for the polynomials g_i. This provides the motivation for a different approach.

Generalizing the efficient algorithm for finding roots of univariate polynomials over polynomial rings proposed by Roth and Ruckenstein in [17], we present an alternative root-finding algorithm below. In this algorithm, a nonzero polynomial is constructed explicitly for the purpose of determining f_{k-i+1}. We will show the correctness of the algorithm by proving that the algorithm computes a set of rational functions which contains all the roots in $L(\rho P)$ of $H(T)$. Also, we can prove that the size of the output set of the algorithm is at most s.

Since the coefficients of $H(T)$ are rational functions and we will evaluate them at the point P, we denote the coefficients h_j of $H(T)$ by $h_j(X)$ and $H(T)$ by $H(X;T)$ in the following algorithm. And we use $[a_1, \ldots, a_k]$ to represent a rational function $a_1\varphi_1+\cdots+a_k\varphi_k$.

```
Algorithm 3.1 Root-Finding Algorithm for Polynomials
over Function Fields
Procedure Finding Roots ($\hat{G}(X;T)$, fixed integer $k$,
integer $i$)
/* Input the nonzero polynomial

        $H(X;T)=h_0(X)+h_1(X)T+\cdots+h_s(X)T^s$

where $h_j(X)\in L(lP)$.
  Input a basis $\{\varphi_1,\ldots,\varphi_k\}$ of $L(G)=L(\rho P)$, and set a
global array $\varphi[1,\ldots,k]=[\varphi_1,\ldots,\varphi_k]$.
  Compute $\hat{G}(X;T)=H(X;\varphi_kT)$.
  A global array $g[1,\ldots,k]$ is assumed.
  The initial call is $(\hat{G}(X;T), k, i=1)$.
*/

Step 1: find a rational function $\phi$ such that

        $\mathrm{ord}_P(\phi)=\min\{\mathrm{ord}_P(\hat{G}^{(j)}(X))\,|\,j=0,1,\ldots,s\}$.

Step 2: $\tilde{G}(X;T)\leftarrow\frac{1}{\phi}\hat{G}(X;T)$.
Step 3: compute the nonzero polynomial $\tilde{G}(P;T)\in\boldsymbol{F}_q[T]$.
Step 4: find all the roots $\alpha$ of $\tilde{G}(P;T)=0$.
Step 5: for each of the distinct roots $\alpha$ of

        $\tilde{G}(P;T)=0$   do {

Step 6:     set $g[i]=\alpha$.
Step 7:     if $i=k$, output $[g_k,g_{k-1},\ldots,g_1]$; else {
Step 8:         set $G(X;T)=\tilde{G}(X;T+\alpha)$.
Step 9:         set $\hat{G}(X;T)=G(X;\frac{\varphi[k-i]}{\varphi[k-i+1]}T)$.
Step 10:         Finding Roots $(\hat{G}(X;T),k,i+1)$.
        }
        }
```
□

It can be proved that Algorithm 3.1 and the Root-Finding Procedure output the same set of rational functions for a given polynomial $H(T)$. In fact, the polynomial equation $\tilde{G}_i(P;T)=0$ constructed for determining f_{k-i+1} using Algorithm 3.1 is equal to the equation $g_i(f_{k-i+1})=0$ in the Root-Finding Procedure. We will see this fact in the example that we will give in Appendix A. Now let us prove the correctness of Algorithm 3.1.

Theorem 3.1: Suppose $H(T)$ is a nonzero polynomial with coefficients in $L(lP)$ as stated in the Root-Finding Problem. Then, Algorithm 3.1 computes a set of rational functions that contains all the roots in $L(\rho P)$ of $H(T)$.

Proof: Let $f=f_1\varphi_1+\cdots+f_k\varphi_k$ be any root in $L(\rho P)$ of $H(X;T)$. It is sufficient to prove that the algorithm can find the coefficients of f. By the algorithm, we have

$$G_1(X;T)=H(X;T)$$

and

$$\hat{G}_1(X;T)=G_1(X;\varphi_kT)$$

Write $\hat{G}_1(X;T)$ as

$$\hat{G}_1(X;T)=\sum_{j=1}^{s}\hat{G}_1^{(j)}(X)T^j.$$

Since ϕ_1 is a rational function such that

$$\mathrm{ord}_P(\phi_1)=\min\left\{\mathrm{ord}_P\left(\hat{G}_1^{(j)}(X)\right)\middle|\,j=0,1,\ldots,s\right\}$$

and

$$\tilde{G}_1(X;T)=\frac{1}{\phi_1}\hat{G}_1(X;T)$$

for the coefficients $\tilde{G}_1^{(j)}(X)$ of $\tilde{G}_1(X;T)$ we have that $\tilde{G}_1^{(j)}(P)\in\boldsymbol{F}_q$ and

$$\left(\tilde{G}_1^{(0)}(P),\tilde{G}_1^{(1)}(P),\ldots,\tilde{G}_1^{(s)}(P)\right)\neq\mathbf{0}.$$

This means that $\tilde{G}_1(P;T)$ is a nonzero polynomial in $\boldsymbol{F}_q[T]$.

From $H(X;\, f(X)) = 0$, we have

$$\hat{G}_1\left(X;\, \frac{f(X)}{\varphi_k(X)}\right) = 0$$

which implies

$$\tilde{G}_1\left(X;\, \frac{f(X)}{\varphi_k(X)}\right) = 0. \tag{3.4}$$

On the other hand, $\mathrm{ord}_P(\frac{\varphi_j}{\varphi_k}) = \rho_k - \rho_j > 0$, for $j = 1, \ldots, k-1$. Therefore, $\frac{\varphi_j}{\varphi_k}(P) = 0$, and

$$\frac{f}{\varphi_k}(P) = f_1\frac{\varphi_1}{\varphi_k}(P) + \cdots + f_{k-1}\frac{\varphi_{k-1}}{\varphi_k}(P) + f_k = f_k.$$

By (3.4), we have

$$\begin{aligned}\tilde{G}_1(P;\, f_k) &= \tilde{G}_1\left(P;\, \frac{f}{\varphi_k}(P)\right)\\ &= \tilde{G}_1\left(X;\, \frac{f(X)}{\varphi_k(X)}\right)(P) = 0.\end{aligned} \tag{3.5}$$

Therefore, f_k is a root of $\tilde{G}_1(P;\, T) = 0$.

Suppose that we have determined coefficients $f_k, \ldots, f_{k-i+1}$. We therefore will have found a polynomial

$$\tilde{G}_i(X;T) = \tilde{G}_i^{(0)}(X) + \tilde{G}_i^{(1)}(X)T + \cdots + \tilde{G}_i^{(s)}(X)T^s$$

such that

$$\tilde{G}_i\left(X;\, \frac{f^{(i)}(X)}{\varphi_{k-i+1}(X)}\right) = 0 \tag{3.6}$$

where $f^{(i)}(X) = f_1\varphi_1 + \cdots + f_{k-i+1}\varphi_{k-i+1}$. Moreover, $\tilde{G}_i(P;\, T)$ is a nonzero polynomial in $\boldsymbol{F}_q[T]$ such that f_{k-i+1} is a solution of $\tilde{G}_i(P;\, T) = 0$. By the algorithm, we have

$$G_{i+1}(X;\, T) = \tilde{G}_i(X;\, T + f_{k-i+1})$$

and

$$\hat{G}_{i+1}(X;\, T) = G_{i+1}\left(X;\, \frac{\varphi_{k-i}}{\varphi_{k-i+1}}T\right).$$

Suppose

$$\hat{G}_{i+1}(X;\, T) = \sum_{j=0}^{s}\hat{G}_{i+1}^{(j)}(X)T^j.$$

Since ϕ_{i+1} is a rational function with

$$\mathrm{ord}_P(\phi_{i+1}) = \min\left\{\mathrm{ord}_P\left(\hat{G}_{i+1}^{(j)}(X)\right)\Big|\, j = 0, 1, \ldots, s\right\}$$

and

$$\tilde{G}_{i+1}(X;\, T) = \sum_{j=0}^{s}\tilde{G}_{i+1}^{(j)}(X)T^j$$

with

$$\tilde{G}_{i+1}^{(j)}(X) = \frac{1}{\phi_{i+1}}\hat{G}_{i+1}^{(j)}(X), \tag{3.7}$$

$\tilde{G}_{i+1}(P;\, T)$ is a nonzero polynomial in $\boldsymbol{F}_q[T]$.

Let

$$\begin{aligned}f^{(i+1)}(X) &= f^{(i)}(X) - f_{k-i+1}\varphi_{k-i+1}(X)\\ &= f_1\varphi_1(X) + \cdots + f_{k-i}\varphi_{k-i}(X).\end{aligned}$$

We have

$$\begin{aligned}\hat{G}_{i+1}\left(X;\, \frac{f^{(i+1)}(X)}{\varphi_{k-i}(X)}\right) &= G_{i+1}\left(X;\, \frac{f^{(i+1)}(X)}{\varphi_{k-i+1}(X)}\right)\\ &= \tilde{G}_i\left(X;\, \frac{f^{(i)}(X)}{\varphi_{k-i+1}(X)}\right) = 0.\end{aligned}$$

Thus,

$$\tilde{G}_{i+1}\left(X;\, \frac{f^{(i+1)}(X)}{\varphi_{k-i}(X)}\right) = 0. \tag{3.8}$$

Since $\frac{f^{(i+1)}}{\varphi_{k-i}}(P) = f_{k-i}$, we have

$$\tilde{G}_{i+1}(P;\, f_{k-i}) = 0. \tag{3.9}$$

Therefore, f_{k-i} is a solution of the nonzero polynomial equation $\tilde{G}_{i+1}(P;\, T) = 0$. By induction, the algorithm can find the coefficients $f_k, f_{k-1}, \ldots, f_1$ of the root f of $H(T)$. □

Lemma 3.2: Suppose α is a solution of multiplicity d of the nonzero polynomial equation $\tilde{G}_i(P;\, T) = 0$, and $\tilde{G}_{i+1}(P;\, T) = 0$ is the corresponding nonzero polynomial equation. Then, $\deg_T \tilde{G}_{i+1}(P;T) \le d$ and $\tilde{G}_{i+1}(P;\, T) = 0$ has at most d roots.

Proof: Let

$$\tilde{G}_i(X;\, T) = \tilde{G}_i^{(0)}(X) + \tilde{G}_i^{(1)}(X)T + \cdots + \tilde{G}_i^{(s)}(X)T^s.$$

Since $\tilde{G}_i(P;\, T) \in \boldsymbol{F}_q[T]$ is a nonzero polynomial, we have

$$\left(\tilde{G}_i^{(0)}(P),\, \tilde{G}_i^{(1)}(P),\, \ldots,\, \tilde{G}_i^{(s)}(P)\right) \in \boldsymbol{F}_q^{s+1} - \{\mathbf{0}\}.$$

Now

$$\begin{aligned}&G_{i+1}(X;\, T)\\ &\quad= G_{i+1}^{(0)}(X) + G_{i+1}^{(1)}(X)T + \cdots + G_{i+1}^{(s)}(X)T^s\\ &\quad= \tilde{G}_i^{(0)}(X) + \tilde{G}_i^{(1)}(X)(T+\alpha) + \cdots + \tilde{G}_i^{(s)}(X)(T+\alpha)^s.\end{aligned}$$

Thus, $G_{i+1}(P;\, T)$ is a polynomial in $\boldsymbol{F}_q[T]$. Since α is a root of $\tilde{G}_i(P;\, T) = 0$ of multiplicity $d, T = 0$ is a root of $G_{i+1}(P;\, T) = 0$ of multiplicity d. So

$$G_{i+1}^{(0)}(P) = G_{i+1}^{(1)}(P) = \cdots = G_{i+1}^{(d-1)}(P)$$

but $G_{i+1}^{(d)}(P) \ne 0$.

Let

$$\hat{G}_{i+1}(X;\, T) = \hat{G}_{i+1}^{(0)}(X) + \cdots + \hat{G}_{i+1}^{(d)}(X)T^d + \cdots + \hat{G}_{i+1}^{(s)}(X)T^s.$$

By Algorithm 3.1

$$\hat{G}_{i+1}^{(j)}(X) = G_{i+1}^{(j)}(X)\left(\frac{\varphi_{k-i}(X)}{\varphi_{k-i+1}(X)}\right)^j, \qquad j = 0, 1, \ldots, s.$$

Because $G_{i+1}^{(d)}(P) \in \boldsymbol{F}_q - \{0\}$, and

$$\mathrm{ord}_P\left(\frac{\varphi_{k-1}}{\varphi_{k-i+1}}\right) = \rho_{k-i+1} - \rho_{k-i} > 0$$

we have

$$\mathrm{ord}_P\left(G_{i+1}^{(d)}(X)\frac{\varphi_{k-1}^d(X)}{\varphi_{k-i+1}^d(X)}\right) = d(\rho_{k-i+1} - \rho_{k-i}) > 0$$

and

$$\begin{aligned}&\mathrm{ord}_P\left(G_{i+1}^{(j)}(X)\frac{\varphi_{k-1}^j(X)}{\varphi_{k-i+1}^j(X)}\right)\\ &\qquad \ge j(\rho_{k-i+1} - \rho_{k-i}) > d(\rho_{k-i+1} - \rho_{k-i}), \qquad j \ge d+1.\end{aligned}$$

Therefore,

$$\begin{aligned}\min\left\{\operatorname{ord}_P\left(\hat{G}_{i+1}^{(j)}\right)\Big| j=0,1,\ldots,s\right\} &= \min\left\{\operatorname{ord}_P\left(\hat{G}_{i+1}^{(j)}\right)\Big| j\le d\right\}\\ &\le d(\rho_{k-i+1}-\rho_{k-i}).\end{aligned}$$

Let ϕ_{i+1} be a rational function with

$$\begin{aligned}\operatorname{ord}_P(\phi_{i+1}) &= \min\left\{\operatorname{ord}_P\left(\hat{G}_{i+1}^{(j)}\right)\Big| j=0,1,\ldots,s\right\}\\ &\le d(\rho_{k-i+1}-\rho_{k-i}).\end{aligned}$$

We have

$$\tilde{G}_{i+1}(X;T)=\frac{\hat{G}_{i+1}^{(0)}}{\phi_{i+1}}+\frac{\hat{G}_{i+1}^{(1)}}{\phi_{i+1}}T+\cdots+\frac{\hat{G}_{i+1}^{(s)}}{\phi_{i+1}}T^s$$

and

$$\operatorname{ord}_P\left(\frac{\hat{G}_{i+1}^{(j)}}{\phi_{i+1}}\right)>0, \qquad \text{for } j\ge d+1$$

i.e., $\frac{\hat{G}_{i+1}^{(j)}}{\phi_{i+1}}(P)=0$ for $j\ge d+1$. Therefore,

$$\deg_T \tilde{G}_{i+1}(P;T)\le d. \qquad \square$$

Corollary 3.3: Suppose $H(T)$ is a polynomial of degree s as in Theorem 3.1. Then, the number of output rational functions generated by Algorithm 3.1 is at most s.

Proof: Suppose $\tilde{G}_1(P;T)=0$ has σ roots, and their multiplicities are $d_{1,1},\ldots,d_{1,\sigma}$, respectively. Clearly, $d_{1,1}+\cdots+d_{1,\sigma}\le s$. When $i=2$, we have σ equations $\tilde{G}_2(P;T)=0$ corresponding to the roots of $\tilde{G}_1(P;T)=0$, and by Lemma 3.2, the degrees of these equations are at most $d_{1,1},\ldots,d_{1,\sigma}$, respectively. So the number of roots of all the equations $\tilde{G}_2(P;T)=0$ is at most $d_{1,1}+\cdots+d_{1,\sigma}\le s$. For $i=1,\ldots,k$, let σ_i and ω_i denote the number of the roots and the sum of the degrees of all the equations $\tilde{G}_i(P;T)=0$, respectively. By induction, we can prove $\sigma_i\le\omega_i\le s$. In particular, the number of output rational functions is $\sigma_k\le s$. $\square$

IV. Efficient List Decoding of AG Codes

Replacing Step 2 in G–S Algorithm by Algorithm 3.1, we get an efficient list-decoding algorithm for AG codes.

```
**Algorithm 4.1** *Efficient List-Decoding Algorithm*
**Implicit Parameters:** Same as in G–S Algorithm.
**Assumptions:** Same as in G–S Algorithm.
**Initialization:** Same as in G–S Algorithm.
**Step 1:** Same as in G–S Algorithm.
**Step 2:** Using Algorithm 3.1, find all roots
        $f \in L((k+g-1)P) = L(\rho P)$
  of the polynomial $H(T)$. For each such $f$, check if
  $f(P_i) = y_i$ for at least $e$ values of $i \in \{1,\ldots,n\}$, and
  if so, include $f$ in the output list.
                                                        □
```

Using [8, Proposition 22 and Theorem 27] and Theorem B.2 that we will give in Appendix B, it is easy to determine the complexity of the list-decoding algorithm above.

Theorem 4.1: Let $\mathcal{X}$ be a nonsingular, absolutely irreducible curve over $\boldsymbol{F}_q$ of genus g, denote by $\mathcal{K}$ the function field of $\mathcal{X}$ over $\boldsymbol{F}_q$. Suppose $C_L(D,\rho P)$ is an AG code over $\mathcal{X}$ of length n, dimension k, and designed distance $d=n-k-g+1$. Then Algorithm 4.1 corrects up to $e<n-\sqrt{n(n-d)}$ errors. The execution of the algorithm needs

$$O(l^3s^3+ks(n^2+s^2+\log^2 s\cdot\log\log s\cdot\log q))$$

operations over $\boldsymbol{F}_q$ and $O(nl^2+ks^2)$ operations over $\mathcal{K}$, where $s=\lfloor\frac{l-g}{n-d}\rfloor$, $l=O(\max\{\frac{gt+n(n-d)}{t^2-n(n-d)},t\})$, and $t>\sqrt{n(n-d)}$.

V. Conclusion

In this correspondence, we have presented an efficient root-finding algorithm for polynomials over function fields, extending Roth and Ruckenstein's *Reconstruct Algorithm* [17] for finding roots of univariate polynomials over polynomial rings. The execution of the algorithm needs $O(ks(n^2+s^2+\log^2 s\cdot\log\log s\cdot\log q))$ operations over $\boldsymbol{F}_q$ and $O(ks^2)$ operations over the function field $\mathcal{K}$. Based on the root-finding algorithm, we gave an efficient list-decoding algorithm for AG codes.

Appendix A
An Example

In this appendix, we give an example of finding the roots of a polynomial over the function field of the Klein quartic over $\boldsymbol{F}_8$.

The Klein quartic over $\boldsymbol{F}_8$ is defined by the following projective equation:

$$X^3Y+Y^3Z+Z^3X=0.$$

The genus of this curve is $g=3$. The Klein quartic has 24 rational points. Three points $Q=(0:1:0)$, $P_1=(1:0:0)$, and $P_2=(0:0:1)$ are defined over $\boldsymbol{F}_2$, the other rational points $P_3,\ldots,P_{23}$ are defined over $\boldsymbol{F}_8$. Consider the linear space $L(mQ)$ with $m>3$. Letting $x=\frac{X}{Z}$ and $y=\frac{Y}{Z}$, it can be proved that $L(mQ)$ has a basis $\{\varphi_1,\ldots,\varphi_{m-2}\}$, where

$$\begin{array}{lll}\varphi_1=1 & \varphi_2=y & \varphi_3=xy\\ \varphi_4=y^2 & \varphi_5=x^2y & \varphi_6=xy^2\end{array}$$

and for $j\ge 3$

$$\varphi_{3j-2}=y^j \qquad \varphi_{3j-1}=x^2y^{j-1} \qquad \varphi_{3j}=xy^j.$$

The rational function x has a pole of order 2 at Q, and y has a pole of order 3 at Q, i.e.,

$$\operatorname{ord}_Q(x)=-2, \quad \text{and} \quad \operatorname{ord}_Q(y)=-3.$$

Let $\rho_j=-\operatorname{ord}_Q(\varphi_j)$. The sequence of nongaps at Q is $\rho_1=0$, $\rho_2=3$, $\rho_3=5$, $\rho_4=6$, $\rho_5=7$, $\rho_6=8$, $\rho_7=9$, $\rho_8=10,\ldots$.

Let ζ be a primitive element of $\boldsymbol{F}_8$, satisfying $\zeta^3+\zeta+1=0$. Consider the following polynomial over the function field of the Klein quartic:

$$\begin{aligned}H(T)=&T^5+\left(\zeta x^2y+\zeta^2y^2+\zeta xy+y+\zeta^6\right)T^4\\ &+\left(\zeta^2x^4y^2+xy^4+x^2y^2+\zeta^2y^3+\zeta xy^2\right.\\ &\quad\left.+\zeta^4x^2y+\zeta^4y^2+\zeta xy+1\right)T^3\\ &+\left(x^6y^3+\zeta x^4y^4+x^5y^3+\zeta^4x^4y^3+\zeta^5x^2y^4\right.\\ &\quad+\zeta^4x^3y^3+\zeta^3x^4y^2+\zeta^4x^2y^3+x^3y^2+\zeta^3xy^3\\ &\quad\left.+\zeta^2y^3+\zeta^3xy^2+x^2y+\zeta y^2+\zeta^6xy+y+\zeta^6\right)T^2\\ &+\left(x^6y^3+\zeta x^4y^4+\zeta^2x^5y^3+\zeta^4x^4y^3+\zeta^5x^2y^4\right.\\ &\quad+\zeta^5x^3y^3+\zeta^4x^4y^2+\zeta^2xy^4+\zeta^2x^2y^3+\zeta^4x^3y^2\\ &\quad\left.+\zeta xy^3+\zeta x^2y^2+xy^2+\zeta^2x^2y+\zeta^6xy\right)T\\ &+\left(x^7y^4+\zeta x^5y^5+x^6y^4+\zeta^4x^5y^4+\zeta^5x^3y^5\right.\\ &\quad+\zeta^4x^4y^4+\zeta^4x^5y^3+\zeta^4x^3y^4+\zeta^3x^4y^3\\ &\quad\left.+\zeta^3x^3y^3+\zeta^2x^3y^2\right).\end{aligned}$$

We will find the roots of $H(T)$ in $L(7Q) = \langle \varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5 \rangle$. Let the roots f have the form $f = f_1\varphi_1 + f_2\varphi_2 + f_3\varphi_3 + f_4\varphi_4 + f_5\varphi_5$. We will use Algorithm 3.1 to find these roots.

By Algorithm 3.1, we have

$$\hat{G}_1(X;\, T) = H(X;\, \varphi_5 T) = H(X;\, x^2 y T).$$

Thus,

$$\begin{aligned}\hat{G}_1(X;\, T) = {} & (x^{10}y^5)T^5 \\ & + \left(\zeta x^{10}y^5 + \zeta^2 x^8 y^6 + \zeta x^9 y^5 + x^8 y^5 + \zeta^6\right) T^4 \\ & + \left(\zeta^2 x^{10} y^5 + x^7 y^7 + x^8 y^5 + \zeta^2 x^6 y^6 + \zeta x^7 y^5 \right. \\ & \quad \left. + \zeta^4 x^8 y^4 + \zeta^4 x^6 y^5 + \zeta x^7 y^4 + x^6 y^3\right) T^3 \\ & + \left(x^{10} y^5 + \zeta x^8 y^6 + x^9 y^5 + \zeta^4 x^8 y^5 + \zeta^5 x^6 y^6 \right. \\ & \quad + \zeta^4 x^7 y^5 + \zeta^3 x^8 y^4 + \zeta^4 x^6 y^5 + x^7 y^4 \\ & \quad + \zeta^3 x^5 y^5 + \zeta^2 x^4 y^5 + \zeta^3 x^5 y^4 + x^5 y^3 \\ & \quad \left. + \zeta x^4 y^4 + \zeta^6 x^5 y^3 + x^4 y^3 + \zeta^6 x^4 y^2\right) T^2 \\ & + \left(x^8 y^4 + \zeta x^6 y^5 + \zeta^2 x^7 y^4 + \zeta^4 x^6 y^4 \right. \\ & \quad + \zeta^5 x^4 y^5 + \zeta^5 x^5 y^4 + \zeta^4 x^6 y^3 \\ & \quad + \zeta^2 x^3 y^5 + \zeta^2 x^4 y^4 + \zeta^4 x^5 y^3 \\ & \quad \left. + \zeta x^3 y^4 + \zeta x^4 y^3 + x^3 y^3 + \zeta^2 x^4 y^2 + \zeta^6 x^3 y^2\right) T \\ & + \left(x^7 y^4 + \zeta x^5 y^5 + x^6 y^4 + \zeta^4 x^5 y^4 + \zeta^5 x^3 y^5 \right. \\ & \quad + \zeta^4 x^4 y^4 + \zeta^4 x^5 y^3 + \zeta^4 x^3 y^4 + \zeta^3 x^4 y^3 \\ & \quad \left. + \zeta^3 x^3 y^3 + \zeta^2 x^3 y^2\right).\end{aligned}$$

Again by Algorithm 3.1, we can take $\phi_1 = x^{10}y^5$. So, $\tilde{G}_1(X;\, T) = \hat{G}_1(X;\, T)/(x^{10}y^5)$, and

$$\begin{aligned}\tilde{G}_1(Q;\, T) &= T^5 + \zeta T^4 + \left(\zeta^2 + 1\right) T^3 + T^2 \\ &= \left(T + \zeta^3\right)^2 T^2 (T + \zeta).\end{aligned}$$

Solving $\tilde{G}_1(Q;\, T) = 0$, we get $f_5 = \zeta^3$, or $f_5 = 0$, or $f_5 = \zeta$.

Now, from $f_5 = \zeta^3$, using Algorithm 3.1 we have

$$G_2(X;\, T) = \tilde{G}_1(X;\, T + \zeta^3)$$

and

$$\hat{G}_2(X;\, T) = G_2(X;\, (\varphi_4/\varphi_5)T) = G_2(X;\, (y/x^2)T).$$

We then get

$$\tilde{G}_2(Q;\, T) = T^2.$$

Solving $\tilde{G}_2(Q;\, T) = 0$, we get $f_4 = 0$.

From $f_4 = 0$, we have

$$G_3(X;\, T) = \tilde{G}_2(X;\, T)$$

and

$$\hat{G}_3(X;\, T) = G_3(X;\, (\varphi_3/\varphi_4)T) = G_3(X;\, (x/y)T)$$

We then get

$$\tilde{G}_3(Q;\, T) = T^2.$$

Solving $\tilde{G}_3(Q;\, T) = 0$, we get $f_3 = 0$.

From $f_3 = 0$, we have

$$G_4(X;\, T) = \tilde{G}_3(X;\, T)$$

and

$$\hat{G}_4(X;\, T) = G_4(X;\, (\varphi_2/\varphi_3)T) = G_4(X;\, (1/x)T)$$

We then get

$$\tilde{G}_3(Q;\, T) = T^2 + T.$$

Solving $\tilde{G}_4(Q;\, T) = 0$, we get $f_2 = 1$ or $f_2 = 0$.

From $f_2 = 1$, we have

$$G_5(X;\, T) = \tilde{G}_4(X;\, T + 1)$$

and

$$\hat{G}_5(X;\, T) = G_5(X;\, (\varphi_1/\varphi_2)T) = G_5(X;\, (1/y)T).$$

We then get $\tilde{G}_3(Q;\, T) = T + \zeta^6$. Solving $\tilde{G}_5(Q;\, T) = 0$, we get $f_1 = \zeta^6$.

From $f_2 = 0$, we have

$$G_5(X;\, T) = \tilde{G}_4(X;\, T)$$

and

$$\hat{G}_5(X;\, T) = G_5(X;\, (\varphi_1/\varphi_2)T) = G_5(X;\, (1/y)T).$$

We then get $\tilde{G}_3(Q;\, T) = T$. Solving $\tilde{G}_5(Q;\, T) = 0$, we get $f_1 = 0$.

So, we obtain two rational functions f with form

$$f = f_1\varphi_1 + f_2\varphi_2 + f_3\varphi_3 + f_4\varphi_4 + f_5\varphi_5.$$

They are

$$\zeta^6 + \varphi_2 + \zeta^3\varphi_5 \quad \text{and} \quad \zeta^3\varphi_5.$$

They are roots in $L(7Q)$ of $H(T)$.

Similarly, for $f_5 = \zeta$, we can obtain another root $1 + \zeta\varphi_3 + \zeta^2\varphi_4 + \zeta\varphi_5$.

Let us consider the case of $f_5 = 0$. By Algorithm 3.1, we get $\tilde{G}_2(Q;\, T) = T^2$. We then get $f_4 = 0$ by solving $\tilde{G}_2(Q;\, T) = 0$. From $f_4 = 0$, we have $\tilde{G}_3(Q;\, T) = T^2$. So, we have $f_3 = 0$. Then from $f_3 = 0$, we get $\tilde{G}_4(Q;\, T) = T^2$. Solving $\tilde{G}_4(Q;\, T) = 0$, we have $f_2 = 0$. Finally, from $f_2 = 0$, we get $\tilde{G}_5(Q;\, T) = 1$. The equation $\tilde{G}_5(Q;\, T) = 0$ has no solution.

Therefore, we have found all the three roots in $L(7Q)$ of $H(T)$. We can graphically represent the root-finding procedure as shown at the bottom of the page. There are three "effective" paths in this graph, which start from $H(f) = 0$ and terminate at $f_1 = \zeta^6$, $f_1 = 0$ and $f_1 = 1$, respectively. Each of them corresponds to a root of $H(T)$.

As a polynomial over the function field of the Klein quartic, $H(T)$ can be factorized as

$$\begin{aligned}H(T) = {} & \left(T + \zeta^3\varphi_5 + \varphi_2 + \zeta^6\right)\left(T + \zeta^3\varphi_5\right) \\ & \cdot \left(T + \zeta\varphi_5 + \zeta^2\varphi_4 + \zeta\varphi_3 + 1\right)\left(T^2 + T + \varphi_3\right).\end{aligned}$$

The last path of the graph at the bottom of the page corresponds to the factor $T^2 + T + \varphi_3$. The fact that $\tilde{G}_5(Q;\, T) = 0$ has no solution shows that $T^2 + T + \varphi_3$ is irreducible over the function field.

Also, we can use the Root-Finding Procedure in Section III to find the roots of $H(T)$. In fact, for each root $f_1\varphi_1 + f_2\varphi_2 + f_3\varphi_3 + f_4\varphi_4 + f_5\varphi_5$, the polynomial g_i that we used to get f_{6-i} by the Root-Finding Procedure is equal to $\tilde{G}_i(Q;\, T)$, $i = 1, \ldots, 5$. When we use the

$$\begin{tikzcd}
 & & & & f_2 = 1 \arrow[r] & f_1 = \zeta^6 \\
 & f_5 = \zeta^3 \arrow[r] & f_4 = 0 \arrow[r] & f_3 = 0 \arrow[ur] \arrow[dr] & & \\
 & & & & f_2 = 0 \arrow[r] & f_1 = 0 \\
H(f) = 0 \arrow[uur] \arrow[r] \arrow[dr] & f_5 = \zeta \arrow[r] & f_4 = \zeta^2 \arrow[r] & f_3 = \zeta \arrow[r] & f_2 = 0 \arrow[r] & f_1 = 1 \\
 & f_5 = 0 \arrow[r] & f_4 = 0 \arrow[r] & f_3 = 0 \arrow[r] & f_2 = 0 \arrow[r] & \text{No solution}
\end{tikzcd}$$

Root-Finding Procedure, we need some operations that reduce $H(f)$ modulo the curve. For example, consider the coefficients of $H(T)$. In the coefficient of T^3, the terms $\zeta^2 x^4 y^2$ and xy^4 have the same pole order at Q. By the affine equation of the Klein quartic

$$x^3 y + y^3 + x = 0$$

we have $xy^4 = x^4y^2 + x^2y$. So, we have $\zeta^2 x^4 y^2 + xy^4 = \zeta^6 x^4 y^2 + x^2 y$, where x^4y^2 and x^2y have different pole orders at Q. So, we can replace $\zeta^2 x^4 y^2 + xy^4$ by $\zeta^6 x^4 y^2 + x^2 y$ to get the unique leading term of the coefficient of T^3.

Appendix B
Complexity Analysis

We evaluate in this appendix the complexity of the root-finding algorithm we proposed in Section III, Algorithm 3.1.

Lemma B.1 [2]: The roots in $\boldsymbol{F}_q$ of a polynomial of degree s can be found in expected time complexity $O((s \log^2 s) \cdot (\log\log s) \cdot \log q)$.

Theorem B.2: Let $\mathcal{X}$ be a nonsingular, absolutely irreducible curve in m-dimensional projective space $\boldsymbol{PF}_q^m$ over the finite field $\boldsymbol{F}_q$, and $\mathcal{K}$ be the function field of $\mathcal{X}$ over $\boldsymbol{F}_q$. Suppose the curve satisfies $q^{m+1} = O(n^2)$, where n is the number of rational points. Given a nonzero polynomial $H(T)$ in $\mathcal{K}[T]$ of degree s returned in Step 1 of the list-decoding algorithm, the execution of the root-finding algorithm, Algorithm 3.1, needs $O(ks(n^2 + s^2 + \log^2 s \cdot \log\log s \cdot \log q))$ operations over $\boldsymbol{F}_q$ and $O(ks^2)$ operations over $\mathcal{K}$.

Proof: From the proof of Corollary 3.3, we see that for every i, the number of polynomials $\hat{G}_i(X; T)$ that arise is less than or equal to σ_{i-1}, the number of roots of all the equations $\tilde{G}_{i-1}(P; T) = 0$. This number is less than or equal to s, the upper bound on the T-degree of every polynomial $\hat{G}_i(X; T)$. So, for every i, the execution of Step 2 needs $O(s^2)$ operations over $\mathcal{K}$. Therefore, the contribution of Step 2 to the overall complexity is $O(ks^2)$ operations over $\mathcal{K}$. Similarly, the contribution of Step 9 to the overall complexity is also $O(ks^2)$ operations over $\mathcal{K}$.

Now consider the complexity of Step 8. From $G_{i+1}(X; T) = \tilde{G}_i(X; T + \alpha)$, we have

$$\begin{aligned} G_{i+1}^{(j)}(X) = \tilde{G}_i^{(j)}(X) &+ \binom{j+1}{1} \alpha \tilde{G}_i^{(j+1)}(X) \\ &+ \binom{j+2}{2} \alpha^2 \tilde{G}_i^{(j+2)}(X) \\ &+ \cdots + \binom{s}{s-j} \alpha^{s-j} \tilde{G}_i^{(s)}(X). \end{aligned}$$

Thus, we require $O(s^2)$ operations over $\boldsymbol{F}_q$ to get each $G_{i+1}(X; T)$. For every $i+1$, $i = 1, \ldots, k-1$, the number of polynomials $G_{i+1}(X; T)$'s is at most s. Therefore, the execution of Step 8 requires $O(ks^3)$ operations over $\boldsymbol{F}_q$.

Next we count the number of operations that will be used to execute Step 4. For $i = 1$, we need to solve a nonzero polynomial equation over $\boldsymbol{F}_q$ of degree s. By Lemma B.1, we need

$$O((s \log^2 s) \cdot (\log\log s) \cdot \log q)$$

operations over $\boldsymbol{F}_q$ to accomplish this. Suppose $\tilde{G}_1(P; T) = 0$ has σ roots, and the multiplicities of the roots are $d_{1,1}, \ldots, d_{1,\sigma}$, respectively. It is clear that every $d_{1,j} \le s$ and $d_{1,1} + \cdots + d_{1,\sigma} \le s$. When $i = 2$, we need to solve σ equations, whose degrees are at most $d_{1,1}, \ldots, d_{1,\sigma}$, respectively. So, we need

$$\begin{aligned} &O((d_{1,1} \log^2 d_{1,1}) \cdot (\log\log d_{1,1}) \cdot \log q) + \cdots \\ &\qquad + O((d_{1,\sigma} \log^2 d_{1,\sigma}) \cdot (\log\log d_{1,\sigma}) \cdot \log q) \end{aligned}$$

operations. Since $\log^2 d_{1,j} \le \log^2 s$, $\log\log d_{1,j} \le \log\log s$, and $d_{1,1} + \cdots + d_{1,\sigma} \le s$, we know that the number of operations for $i = 2$ is upper-bounded by $O((s \log^2 s) \cdot (\log\log s) \cdot \log q)$. By induction, for every i, the execution of Step 4 requires

$$O((s \log^2 s) \cdot (\log\log s) \cdot \log q)$$

operations. Therefore, Step 4 in the algorithm can be completed in $O((ks \log^2 s) \cdot (\log\log s) \cdot \log q)$ operations over $\boldsymbol{F}_q$.

Finally, we consider the complexity associated with the execution of Steps 1 and 3. In these steps, we need to evaluate rational functions at P. To get $\tilde{G}_i(P; T)$, we need to compute $\tilde{G}_i^{(j)}(P)$ for $j = 0, \ldots, s$. It is known that there is an open neighborhood U with $P \in U \subseteq \mathcal{X}$, and $N_i^{(j)}, D_i^{(j)} \in \boldsymbol{F}_q[X_1, \ldots, X_{m+1}]$, such that on U

$$\tilde{G}_i^{(j)}(X) = \frac{N_i^{(j)}(X_1, \ldots, X_{m+1})}{D_i^{(j)}(X_1, \ldots, X_{m+1})}$$

where $N_i^{(j)}$ and $D_i^{(j)}$ are homogeneous polynomials and $D_i^{(j)}$ is nowhere zero on U. As a function, $X_j^q = X_j$ over $\boldsymbol{F}_q$, so every term of $N_i^{(j)}$ and $D_i^{(j)}$ has degree at most $(m+1)(q-1)$. Therefore, we require $O(q^{m+1})$ operations to compute $\tilde{G}_i^{(j)}(P)$. Since $q^{m+1} = O(n^2)$, $O(n^2)$ operations are needed to compute every $\tilde{G}_i^{(j)}(P)$. By Lemma 3.2, if the root of $\tilde{G}_{i-1}(P; T)$ that we used to construct $\tilde{G}_i(P; T)$ has multiplicity d, where $d \le s$, then $\tilde{G}_i^{(j)}(P) = 0$ for $j \ge d+1$. So $O(dn^2)$ operations over $\boldsymbol{F}_q$ are needed to get $\tilde{G}_i(P; T)$. It follows that the execution of Step 3 needs $O(ksn^2)$ operations over $\boldsymbol{F}_q$. We know that $\mathrm{ord}_P(\hat{G}_i^{(j)}(X)) \ge 0$ for $j = 0, 1, \ldots, s$. Referring again to Lemma 3.2

$$\min\{\mathrm{ord}_P(\hat{G}_i^{(j)}) \mid j = 0, 1, \ldots, s\} = \min\{\mathrm{ord}_P(\hat{G}_i^{(j)}) \mid j \le d\}.$$

Now, for every $j \le d$, divide $\hat{G}_i^{(0)}(X), \hat{G}_i^{(1)}(X), \ldots, \hat{G}_i^{(d)}(X)$ by $\hat{G}_i^{(j)}(X)$. If

$$\left(\frac{\hat{G}_i^{(0)}}{\hat{G}_i^{(j)}}(P), \frac{\hat{G}_i^{(1)}}{\hat{G}_i^{(j)}}(P), \ldots, \frac{\hat{G}_i^{(d)}}{\hat{G}_i^{(j)}}(P) \right) \in \boldsymbol{F}_q^{d+1} - \{\boldsymbol{0}\}$$

then we can take $\phi_i = \hat{G}_i^{(j)}(X)$. So, for every $\hat{G}_i(X; T)$, we need $O(d^2)$ operations over $\mathcal{K}$ and $O(dn^2)$ operations over $\boldsymbol{F}_q$ to determine ϕ_i. Thus, the execution of Step 1 needs $O(ks^2)$ operations over $\mathcal{K}$ and $O(ksn^2)$ operations over $\boldsymbol{F}_q$.

In summary, the execution of Algorithm 3.1 needs

$$O(ks(n^2 + s^2 + \log^2 s \cdot \log\log s \cdot \log q))$$

operations over $\boldsymbol{F}_q$ and $O(ks^2)$ operations over $\mathcal{K}$. □

We remark that the assumption of $q^{m+1} = O(n^2)$ in Theorem B.2 is reasonable. In fact, a large class of curves, including many well-known curves, has this property. For examples, the Klein quartic in $\boldsymbol{PF}_8^2$ has $n = 24$ rational points, the plane Hermitian curve in $\boldsymbol{PF}_{q^2}^2$ has $n = q^3$ affine rational points [3], the Garcia–Stichtenoth curve [7] in $\boldsymbol{P}F_{q^2}^m$ has at least $(q^2-1)q^{m-1}$ rational points. On the other hand, it is easy to see that there exist curves that do not satisfy this assumption. This means that the complexity estimate in Theorem B.2 is not applicable for those curves. However, Algorithm 3.1 works for any nonsingular, absolutely irreducible curve defined over $\boldsymbol{F}_q$ and, for a fixed curve, we are always able to get a similar estimate of complexity.

Acknowledgment

The authors wish to thank T. Høholdt for his interest in this work and for sending the papers [1], [9], [13] and [15]. They also wish to thank M. K. Cheng, R. Roth, M. Shokrollahi, and M. Sudan for useful discussions. Finally, they are indebted to the anonymous referees whose thoughtful criticisms and valuable suggestions helped to improve the quality of this correspondence.

References

[1] D. Augot and L. Pecquet, "An alternative to factorization: A speedup for SUDAN's decoding algorithm and its generalization to algebraic-geometric codes," INRIA, France, Rapport de recherche, no. 3532, 1998.

[2] M. Ben-Or, "Probabilistic algorithms in finite fields," in *Proc. 22nd Annu. IEEE Symp. Foundations of Computer Science*, 1981, pp. 394–398.
[3] I. Blake, C. Heegard, T. Høholdt, and V. Wei, "Algebraic-geometry codes," *IEEE Trans. Inform. Theory*, vol. 44, pp. 2596–2618, Oct. 1998.
[4] P. Elias, "List decoding for noisy channel," Res. Lab. Electron., MIT, Cambridge, MA, Tech. Rep. 335, , 1957.
[5] G.-L. Feng and T. R. N. Rao, "Decoding of algebraic geometric codes up to the designed minimum distance," *IEEE Trans. Inform. Theory*, vol. 39, pp. 37–45, Jan. 1993.
[6] S.-H. Gao and M. Shokrollahi, "Computing roots of polynomials over function fields of curves," preprint, 1998.
[7] A. Garcia and H. Stichtenoth, "A tower of Artin–Schreier extensions of function fields attaining the Drinfeld–Vladut bound," *Invent. Math.*, vol. 121, pp. 211–222, 1995.
[8] V. Guruswami and M. Sudan, "Improved decoding of Reed–Solomon and algebraic-geometry codes," *IEEE Trans. Inform. Theory*, vol. 45, pp. 1757–1767, Sept. 1999.
[9] T. Høholdt and R. Nielsen, "Decoding Hermitian codes with Sudan's algorithm," in *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (Lecture Notes in Computer Science)*, N. Fossorier, H. Imei, S. Lin, and A. Pole, Eds. Berlin, Germany: Springer-Verlag, 1999, vol. 1719, pp. 260–270.
[10] T. Høholdt and R. Pellikaan, "On the decoding of algebraic-geometric codes," *IEEE Trans. Inform. Theory*, vol. 41, pp. 1589–1614, Nov. 1995.
[11] J. Justesen, K. J. Larsen, H. E. Jensen, A. Havemose, and T. Høholdt, "Construction and decoding of a class of algebraic geometry codes," *IEEE Trans. Inform. Theory*, vol. 35, pp. 811–821, July 1989.
[12] R. Lidl and H. Niederreiter, *Finite Fields*. Reading, MA: Addison-Wesley, 1983.
[13] R. Nielsen and T. Høholdt, "Decoding Reed–Solomon codes beyond half the minimum distance," in *Coding Theory, Cryptography and Related Areas*, J. Buchmann, T. Høholdt, T. Stichtenoth, and H. Tapia-Recillas, Eds. Berlin, Germany: Springer-Verlag, 2000, pp. 221–236.
[14] V. Olshevsky and M. Shokrollahi, "A displacement approach to efficient decoding of algebraic-geometric codes," preprint.
[15] L. Pecquet, "An algorithm to get some factors of bivariate polynomials, without factoring," preprint.
[16] M. O. Rabin, "Probabilistic algorithms in finite fields," *SIAM J. Comput.*, vol. 9, pp. 273–280, 1980.
[17] R. Roth and G. Ruckenstein, "Efficient decoding of Reed–Solomon codes beyond half the minimum distance," *IEEE Trans. Inform. Theory*, vol. 46, pp. 246–257, Jan. 2000.
[18] M. Shokrollahi and H. Wasserman, "List decoding of algebraic-geometric codes," *IEEE Trans. Inform. Theory*, vol. 45, pp. 432–437, Mar. 1999.
[19] M. Sudan, "Decoding of Reed-Solomon codes beyond the error-correction bound," *J. Complexity*, vol. 13, pp. 180–193, 1997.
[20] J. M. Wozencraft, "List decoding," *Quarterly Progr. Rept.*, vol. 48, pp. 90–95, Jan. 15, 1958. Res. Lab. Electron., MIT.
[21] X.-W. Wu and P. H. Siegel, "Fast computation of roots of polynomials over function fields and fast list decoding of algebraic-geometric codes," in *Proc. Int. Symp. Information Theory (ISIT'2000)*, Sorrento, Italy, June 2000, p. 478.

Single-Track Circuit Codes

Alain P. Hiltgen and Kenneth G. Paterson, *Member, IEEE*

***Abstract*—Single-track circuit codes (STTCs) are circuit codes with codewords of length n such that all the n tracks which correspond to the n distinct coordinates of the codewords are cyclic shifts of the first track. These codes simultaneously generalize single-track Gray codes and ordinary circuit codes. They are useful in angular quantization applications in which error detecting and/or correcting capabilities are needed. A parameter k, called the spread of the code, measures the strength of this error control capability. We consider the existence of STCCs for small lengths $n \leq 17$ and spreads $k \leq 6$, constructing some optimal and many good examples. We then give a general construction method for STCCs which makes use of ordinary circuit codes. We use this construction to construct examples of codes with 360 and 1000 codewords which are of practical importance. We also use the construction to prove a general result on the existence of STCCs for general spreads.**

***Index Terms*—Absolute angle measurement, circuit code, digital encoding, error correction, Gray code, quantization, single track, snake-in-the-box code.**

I. Introduction

A length-n Gray code is simply a cyclic list of distinct binary n-tuples, called the codewords, with the property that any two adjacent codewords differ in exactly one component. A common use of Gray codes is in reducing quantization errors in various types of analog-to-digital conversion systems [11], [12]. They have also found applications in many other areas of coding and computing science—see the introduction to [18] for a list of references.

Spread-k circuit codes are a generalization of Gray codes: they can be thought of as being Gray codes having additional error-detecting capability. For $k \geq 1$, a spread-k code is defined to be a Gray code in which two words of the code either lie at most $k-1$ positions apart in the list of codewords or differ in at least k components. Thus, a spread-1 code is just a Gray code. Spread–2 codes are more commonly known as snake-in-the-box codes. Circuit codes have a long history (see [1] and the references cited there), and many optimal codes and general constructions for families of codes are known: these results are summarized in Section III below.

As an example of the use of circuit codes in analog-to-digital conversion, a length-n, spread-k circuit code C can be used to record the absolute angular positions of a rotating wheel by encoding (e.g., optically) the codewords of C in sectors on n concentrically arranged tracks. Then n reading heads, mounted radially across the tracks suffice to recover the codewords. The number of codewords in C determines the accuracy with which angles can be resolved. Quantization errors are minimized by using a Gray encoding while errors resulting from equipment malfunction can be dealt with using the spread capability of the code: any error of weight $r < k$ either results in an angle precisely r sectors away from the correct sector or leads to a word W that does not lie in C (so that the error is detectable). In the latter case, if $2r < k$, then the word W' in C that is closest to W in Hamming distance is, in turn, at most distance r from the correct codeword. The resulting angular error is at most r sectors. In this way, errors of weight up to $\lfloor \frac{k-1}{2} \rfloor$ can be "partially corrected."

Manuscript received July 13, 2000; revised March 15, 2001.
A. P. Hiltgen is with UBS AG, IT Security, CH-8098, Zurich, Switzerland.
K. G. Paterson is with Hewlett-Packard Laboratories, Stoke Gifford, Bristol BS34 8QZ, U.K. (e-mail: kp@hplb.hpl.hp.com).
Communicated by R. M. Roth, Associate Editor for Coding Theory.
Publisher Item Identifier S 0018-9448(01)06225-3.