INFORMATION AND CONTROL **27**, 87–99 (1975)

# A Method for Solving Key Equation for Decoding Goppa Codes*

YASUO SUGIYAMA,[†] MASAO KASAHARA,[††] SHIGEICHI HIRASAWA,[†] AND TOSHIHIKO NAMEKAWA[††]

*Department of Communication Research and Development,*[†]

*Communication Equipment Works, Mitsubishi Electric Corporation, 80 Nakano, Minamishimizu, Amagasaki, Hyogo, 661 Japan, and*

*Department of Communication Engineering,*[††] *Faculty of Engineering, Osaka University, Yamada-Kami, Suita, Osaka, 565 Japan*

In this paper we show that the key equation for decoding Goppa codes can be solved using Euclid's algorithm. The division for computing the greatest common divisor of the Goppa polynomial $g(z)$ of degree $2t$ and the syndrome polynomial is stopped when the degree of the remainder polynomial is less than or equal to $t - 1$. The error locator polynomial is proved the multiplier polynomial for the syndrome polynomial multiplied by an appropriate scalar factor. The error evaluator polynomial is proved the remainder polynomial multiplied by an appropriate scalar factor. We will show that the Euclid's algorithm can be modified to eliminate multiplicative inversion, and we will evaluate the complexity of the inversionless algorithm by the number of memories and the number of multiplications of elements in $GF(q^m)$. The complexity of the method for solving the key equation for decoding Goppa codes is a few times as much as that of the Berlekamp–Massey algorithm for BCH codes modified by Burton. However the method is straightforward and can be applied for solving the key equation for any Goppa polynomial.

## 1. INTRODUCTION

V. D. Goppa (1970, 1971, 1972) introduced a wide class of error correcting codes which includes BCH codes and Srivastava codes as subclasses. The Goppa codes have a remarkable property that almost all long irreducible

* The contents of this paper were presented orally at the 1974 International Symposium on Information Theory, which was held at Notre Dame, Indiana, October 27–31, 1974.

643/27/1-7

Goppa codes satisfy the Varsharmov–Gilbert bound. Let us define the following symbols.

$q$ = prime power,

$m$ = integer,

$g(z)$ = polynomial over $GF(q^m)$,

$L$ = subset of elements of $GF(q^m)$ excluding roots of $g(z)$,

$n$ = code length (number of elements in $L$),

$\boldsymbol{a}$ = vector of dimension $n$,

$a_i$ = $i$th coordinate of $\boldsymbol{a}$ and element in $GF(q)$,

$\alpha_i$ = element in $L$.

The Goppa code over $GF(q)$ with code length $n$ is defined as a set of vectors $\boldsymbol{a}$ that satisfy

$$\sum_{i=1}^{n} \frac{a_i}{z - \alpha_i} \equiv 0 \qquad \bmod g(z). \tag{1}$$

In congruence (1), we assume all $\alpha_i$'s are distinct. The number of check symbols of the code is at most $m \deg g$, where $\deg g$ denotes the degree of $g(z)$. The minimum distance of the code is at least $(\deg g + 1)$. The polynomial §$(z)$ is called Goppa polynomial.

The algebraic decoding of the code can be done as follows. Let $e_i$ and $r_i$ be the $i$th coordinates of an error vector and a received word vector, respectively, and these be elements in $GF(q)$. Let $E$ be the set of integers such that $e_i \neq 0$. Then the syndrome polynomial $S(z)$, the error locator polynomial $\sigma(z)$, and the error evaluator polynomial $\eta(z)$ are given by

$$S(z) = -\sum_{i=1}^{n} r_i \frac{g(z) - g(\alpha_i)}{z - \alpha_i} g^{-1}(\alpha_i)$$

$$\sigma(z) = \prod_{i \in E} (z - \alpha_i)$$

and

$$\eta(z) = \sum_{i \in E} e_i \prod_{\substack{i' \in E \\ i' \neq i}} (z - \alpha_{i'}).$$

From these three polynomials we see the following important relation holds

$$S(z) \equiv \frac{\eta(z)}{\sigma(z)} \qquad \bmod g(z). \tag{2}$$

When once $\sigma(z)$ and $\eta(z)$ become known for given $S(z)$, the error locations are given by the roots of $\sigma(z)$ and the error value $e_i$ for the error location $i (i \in E)$ is given by $\eta(\alpha_i)/\sigma'(\alpha_i)$ without knowing other error locations, where $\sigma'(z)$ is the formal derivative of the polynomial $\sigma(z)$. Therefore our interests are concentrated upon solving the congruence (2). We notice that $\eta(z)$ and $\sigma(z)$ are relatively prime and so are $\sigma(z)$ and $g(z)$. Berlekamp (1973) calls the congruence (2) the key equation for decoding Goppa codes, and describes that given $g(z)$ and $S(z)$ the decoder's problem is to find low-degree polynomials $\sigma(z)$ and $\eta(z)$ that satisfy (2) and no method of comparable simplicity with his iterative decoding algorithm (Berlekamp, 1968) for BCH codes is known for solving the problem.

For binary case, the numerator of the rational function of the left-hand side of the congruence (1) is divisible by a polynomial $\bar{g}(z)$ where $\bar{g}(z)$ is the least degree polynomial which is a perfect square and such that $g(z)$ divides $\bar{g}(z)$. The number of check symbols of the binary codes is at most $m \deg g$ and the minimum distance is at least $(\deg \bar{g} + 1)$. The syndrome polynomial can be defined by

$$\bar{S}(z) = \sum_{i=1}^{n} r_i \frac{\bar{g}(z) - \bar{g}(\alpha_i)}{z - \alpha_i} \bar{g}^{-1}(\alpha_i).$$

The key equation is given by

$$\bar{S}(z) \equiv \frac{\sigma'(z)}{\sigma(z)} \qquad \mod \bar{g}(z). \tag{3}$$

Setting $\deg g = 2t$ for nonbinary case and $\deg \bar{g} = 2t$ for binary case, the key equation of (2) for nonbinary case and that of (3) for binary case can be treated in the same manner.

In this paper, we describe a method for solving the key equation for decoding Goppa codes using Euclid's algorithm. In Section 2, we describe some of the properties of Euclid's algorithm that are particularly related to the method. In Section 3, we discuss the method. In Section 4, we show that the method can be modified to eliminate multiplicative inversion. In Section 5, the complexity of the inversionless algorithm is evaluated by the number of memories and the number of multiplications of elements in $GF(q^m)$.

## 2. Properties of Euclid's Algorithm

We describe some of the properties of Euclid's algorithm that are particularly related to the method for solving the key equation for decoding Goppa codes. The Euclid's algorithm for computing the greatest common divisor of two polynomials $r_{-1}(z)$ and $r_0(z)$ can be done iteratively as follows, where we assume $\deg r_{-1} > \deg r_0$.

$$\begin{array}{lll}
r_{-1}(z) = q_1(z)\, r_0(z) + r_1(z), & \deg r_{-1} = \deg q_1 + \deg r_0, & \deg r_0 > \deg r_1, \\
r_0(z) = q_2(z)\, r_1(z) + r_2(z), & \deg r_0 = \deg q_2 + \deg r_1, & \deg r_1 > \deg r_2, \\
\quad\vdots & \quad\vdots & \quad\vdots \\
r_{i-2}(z) = q_i(z)\, r_{i-1}(z) + r_i(z), & \deg r_{i-2} = \deg q_i + \deg r_{i-1}, & \deg r_{i-1} > \deg r_i, \\
\quad\vdots & \quad\vdots & \quad\vdots \quad .
\end{array} \tag{4}$$

We call $q_i(z)$ a quotient polynomial and $r_i(z)$ a remainder polynomial at the $i$th iteration for $i = 1, 2,\ldots$. The division terminates at the $h$th iteration when $r_h(z) = 0$. The greatest common divisor can be obtained uniquely by $r_{h-1}(z)$ multiplied by a scalar factor.

The relation (4) can be rewritten in the matrix form as follows:

$$\begin{pmatrix} r_{i-2}(z) \\ r_{i-1}(z) \end{pmatrix} = \begin{pmatrix} q_i(z) & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} r_{i-1}(z) \\ r_i(z) \end{pmatrix}. \tag{5}$$

Defining polynomials $U_i(z)$ and $V_i(z)$ as follows,

$$U_i(z) = q_i(z)\, U_{i-1}(z) + U_{i-2}(z) \tag{6}$$

and

$$V_i(z) = q_i(z)\, V_{i-1}(z) + V_{i-2}(z),$$

where $U_0(z) = 1$, $U_{-1}(z) = 0$, $V_0(z) = 0$ and $V_{-1}(z) = 1$, we have

$$\begin{pmatrix} U_i(z) & U_{i-1}(z) \\ V_i(z) & V_{i-1}(z) \end{pmatrix} = \begin{pmatrix} q_1(z) & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} q_2(z) & 1 \\ 1 & 0 \end{pmatrix} \cdots \begin{pmatrix} q_i(z) & 1 \\ 1 & 0 \end{pmatrix}. \tag{7}$$

From (5) and (7), we obtain

$$\begin{pmatrix} r_{-1}(z) \\ r_0(z) \end{pmatrix} = \begin{pmatrix} U_i(z) & U_{i-1}(z) \\ V_i(z) & V_{i-1}(z) \end{pmatrix} \begin{pmatrix} r_{i-1}(z) \\ r_i(z) \end{pmatrix}.$$

The determinant of the matrix of the left-hand side of (7) is $(-1)^i$. We have

$$\begin{pmatrix} r_{i-1}(z) \\ r_i(z) \end{pmatrix} = (-1)^i \begin{pmatrix} V_{i-1}(z) & -U_{i-1}(z) \\ -V_i(z) & U_i(z) \end{pmatrix} \begin{pmatrix} r_{-1}(z) \\ r_0(z) \end{pmatrix}.$$

Thus the remainder polynomial $r_i(z)$ at the $i$th iteration can be related to the polynomials $r_{-1}(z)$ and $r_0(z)$ as follows,

$$r_i(z) = (-1)^i \{-V_i(z)\, r_{-1}(z) + U_i(z)\, r_0(z)\}. \tag{8}$$

We call $U_i(z)$ the multiplier polynomial for the polynomial $r_0(z)$ at the $i$th iteration. From this relation it is evident that $r_{-1}(z)$ divides the polynomial $\{(-1)^i U_i(z) r_0(z) - r_i(z)\}$. Therefore we obtain the relation

$$r_i(z) \equiv (-1)^i U_i(z) r_0(z) \qquad \mathrm{mod}\, r_{-1}(z). \tag{9}$$

We notice that this relation is quite analogous to the key equation for decoding Goppa codes.

The other properties of Euclid's algorithm that are related to the method for solving the key equation for decoding Goppa codes are summarized as follows. From (4), it is obvious that

$$\deg r_i < \deg r_{i-1}. \tag{10}$$

From (6), $\deg U_i = \sum_{j=1}^{i} \deg q_j$. Since $\deg r_{i-1} = \deg r_{-1} - \sum_{j=1}^{i} \deg q_j$ from (4), it is clear that

$$\deg U_i = \deg r_{-1} - \deg r_{i-1}. \tag{11}$$

From (7), we obtain

$$U_i(z) V_{i-1}(z) - U_{i-1}(z) V_i(z) = (-1)^i. \tag{12}$$

The polynomials $U_i(z)$ and $V_i(z)$ are proved relatively prime.

## 3. Method for Solving the Key Equation

We summarize our problem as follows.

Problem. Given the Goppa polynomial $g(z)$ of degree $2t$ and the syndrome polynomial $S(z)$ of degree $< 2t$ which belongs to a set $F$, find a pair of a monic polynomial $\sigma_s(z)$ of degree $\leqslant t$ and a polynomial $\eta_s(z)$ of degree $< t$ that are relatively prime and satisfy

$$\sigma_s(z) S(z) \equiv \eta_s(z) \qquad \mathrm{mod}\, g(z), \tag{13}$$

where the set $F$ is defined as the set of $\sum_{i=1}^{t} (q-1)^i \binom{n}{i}$ syndrome polynomials corresponding to error patterns with the number of errors $e$ actually occured, $1 \leqslant e \leqslant t$.

*Comment.* When $\sigma_s(z)$ and $\eta_s(z)$ are not necessarily relatively prime and $\sigma_s(z)$ is not necessarily monic, the polynomials $\sigma_s(z)$ and $\eta_s(z)$ satisfy

$$\begin{aligned} \sigma_s(z) &= \mu(z)\,\sigma(z) \\ \eta_s(z) &= \mu(z)\,\eta(z), \end{aligned} \tag{14}$$

where the polynomials $\sigma(z)$ and $\eta(z)$ are the error locator polynomial and the error evaluator polynomial which are corresponding to the given syndrome polynomial $S(z)$ and the polynomial $\mu(z)$ is an appropriate polynomial.

*Proof.* From the relations

$$\sigma_s(z)\,S(z) \equiv \eta_s(z) \qquad \mod g(z)$$

and

$$\sigma(z)\,S(z) \equiv \eta(z) \qquad \mod g(z), \tag{15}$$

we obtain the relation

$$\sigma_s(z)\,\eta(z) \equiv \sigma(z)\,\eta_s(z) \qquad \mod g(z).$$

Since the degrees of $\sigma_s(z)\,\eta(z)$ and $\sigma(z)\,\eta_s(z)$ are less than the degree of $g(z)$, we obtain the relation

$$\sigma_s(z)\,\eta(z) = \sigma(z)\,\eta_s(z).$$

Since $\sigma(z)$ and $\eta(z)$ are relatively prime, $\sigma(z)$ must divide $\sigma_s(z)$ and $\eta(z)$ must divide $\eta_s(z)$. We obtain the relation (14). Q.E.D.

It is clear that there exists a unique solution of the problem. The solution $\sigma_s(z)$ and $\eta_s(z)$ of the problem are the error locator polynomial $\sigma(z)$ and error evaluator polynomial $\eta(z)$ which correspond to the given syndrome polynomial $S(z)$.

We will describe several lemmata.

LEMMA 1. *The degree of the syndrome polynomial $S(z)$ satisfies*

$$\deg S \geqslant 2t - \deg \sigma \geqslant t.$$

*Proof.* We assume $\deg S < 2t - \deg \sigma$. Then from (15) we obtain $\deg \eta = \deg S + \deg \sigma$. But from the definitions of $\sigma(z)$ and $\eta(z)$, $\deg \eta < \deg \sigma$. Therefore $\deg S \geqslant 2t - \deg \sigma \geqslant t$.

LEMMA 2. *Let $p(z)$ be the greatest common divisor of $g(z)$ and $S(z)$. Then $p(z)$ divides $\eta(z)$ and* $\deg p \leqslant \deg \eta \leqslant t - 1$.

*Proof.* From (15), we can easily derive the relation.

We have the following fundamental theorem for solving the key equation for decoding Goppa codes, using Euclid's algorithm.

THEOREM. *Set* $r_{-1}(z) = g(z)$ *and* $r_0(z) = S(z)$. *Start the divisions* (4) *of the Euclid's algorithm for computing the greatest common divisor of* $r_{-1}(z)$ *and* $r_0(z)$. *Stop the divisions when the degrees of the remainder polynomials* $r_i(z)$, $(i = 0, 1, 2, \ldots)$, *satisfy* $\deg r_{k-1} \geqslant t$ *and* $\deg r_k \leqslant t - 1$ *for some k. Then the solution is given by*

$$\begin{aligned} \eta_s(z) &= (-1)^k \,\delta r_k(z) \\ \sigma_s(z) &= \delta U_k(z), \end{aligned} \tag{16}$$

*where* $\delta$ *is a nonzero constant which makes* $\sigma_s(z)$ *monic.*

*Proof.* The proof is done according to the following steps.

(1) We show that the number of iterations $k$ when $\deg r_{k-1} \geqslant t$ and $\deg r_k \leqslant t - 1$ are satisfied is unique. We obtain $\deg r_0 = \deg S \geqslant t$ from Lemma 1. There exists the number of iteration $i$ that satisfies $\deg p \leqslant \deg r_i \leqslant t - 1$. From (10), it is clear that the sequence of $\deg r_i$ $(i = 0, 1, 2 \ldots)$ monotonically decreases as $i$ increases. We conclude the number of iterations $k$ is unique.

(2) Since $U_k(z)\, S(z) = (-1)^k r_k(z) \mod g(z)$ is satisfied from (9), it is clear that $\sigma_s(z)$ and $\eta_s(z)$ given by (16) satisfy the relation $\sigma_s(z)\, S(z) = \eta_s(z) \mod g(z)$.

(3) Since $\deg r_k \leqslant t - 1$ is satisfied, $\eta_s(z)$ given by (16) satisfies the relation $\deg \eta_s \leqslant t - 1$.

(4) From (11), it is clear that $\deg U_k = \deg g - \deg r_{k-1}$ holds. Since $\deg r_{k-1} \geqslant t$ is satisfied, $\sigma_s(z)$ given by (16) satisfies the relation $\deg \sigma_s \leqslant t$.

(5) We show that $\sigma_s(z)$ and $\eta_s(z)$ are relatively prime. We assume that $\sigma_s(z)$ and $\eta_s(z)$ are not relatively prime. From (1), (2), (3), and (4) of this proof and the Comment, $\sigma_s(z)$ and $\eta_s(z)$ are given by $\sigma_s(z) = \mu(z)\,\sigma(z)$ and $\eta_s(z) = \mu(z)\,\eta(z)$. From (8) and (15), we obtain the relations

$$\mu(z)\,\eta(z) = \mu(z)\,\sigma(z)\,S(z) - \delta V_k(z)\,g(z)$$
$$\eta(z) = \sigma(z)\,S(z) + \psi(z)\,g(z),$$

where $\psi(z)$ is an appropriate polynomial. From the relations it is evident that $\mu(z)$ divides $V_k(z)$. The polynomial $\mu(z)$ also divides $U_k(z)$. But this is contradictory to (12) which implies $U_k(z)$ and $V_k(z)$ are relatively prime. We conclude $\sigma_s(z)$ and $\eta_s(z)$ are relatively prime.

We have the following corollary about the number of iterations.

(6) The factor $\delta$ makes $\sigma_s(z)$ monic. Therefore $\sigma_s(z)$ and $\eta_s(z)$ given by (16) are the solution of the problem. Q.E.D.

COROLLARY. *Let $e$ be the number of errors actually occurred and let $k$ be the number of iterations of the algorithm described in the theorem, then $k \leqslant e$.*

*Proof.* Since the number of errors actually occured is $e$, $\deg \sigma = e$. From (6) and (16), $\deg \sigma_s = \sum_{i=1}^{k} \deg q_i$. Since $\deg q_i \geqslant 1$ for all $i$, we conclude $k \leqslant e$.

## 4. Inversionless Algorithm

Euclid's algorithm necessitates the divisions of elements in Galois Field. However it is more desirable to eliminate multiplicative inversion. Following the Burton's method (Burton, 1971) that has modified the Berlekamp–Massey algorithm (Berlekamp, 1968; Massey, 1969) for decoding BCH codes to eliminate inversion, we also modify our method which uses Euclid's algorithm for decoding Goppa codes.

Let us consider two polynomials $x(z)$ and $y(z)$

$$x(z) = \sum_{i=0}^{d_x} x_i z^i,$$

$$y(z) = \sum_{i=0}^{d_y} y_i z^i,$$

where $d_x$ and $d_y$ are the degrees of $x(z)$ and $y(z)$, respectively, and we assume $d_x > d_y$. First, we compute the following polynomial $x^{(1)}(z)$

$$x^{(1)}(z) = y_{d_y} x(z) - x_{d_x} z^{d_x - d_y} y(z).$$

Then $\deg x^{(1)} < d_x$ is satisfied. We compute the polynomial $x^{(2)}(z)$

$$x^{(2)}(z) = y_{d_y} x^{(1)}(z) - x^{(1)}_{d_x - 1} z^{d_x - d_y - 1} y(z),$$

where $x^{(i-1)}_{d_x-1}$ is the coefficient of $z^{d_x+1}$ of $x^{(1)}(z)$ and it can be zero. In the similar manner, we obtain

$$x^{(i)}(z) = y_{d_y} x^{(i-1)}(z) - x^{(i-1)}_{d_x-i+1} z^{d_x-d_y-i+1} y(z),$$

where $x^{(i-1)}_{d_x-i+1}$ is the coefficient of $z^{d_x-i+1}$ of $x^{(i+1)}(z)$. After having calculated $x^{(i)}(z)$ for $i = d_x - d_y + 1$, we obtain the relation

$$\nu x(z) = Q(z)\, y(z) + R(z),$$

where

$$\begin{aligned} Q(z) &= \sum_{i=0}^{d_x-d_y} x^{(d_x-d_y-i)}_{d_y+i} (y_{d_y})^i\, z^i, \\ R(z) &= x^{(d_x-d_y+1)}(z), \\ \nu &= (y_{d_y})^{d_x-d_y+1}, \\ \deg Q &= d_x - d_y\,, \\ \deg R &< d_y\,. \end{aligned}$$

Using the algorithm at each step of Euclid's algorithm, we can modify our algorithm to eliminate inversion. Corresponding to (4), we obtain

$$\begin{array}{lll} \nu_1 \bar{r}_{-1}(z) = \bar{q}_1(z)\, \bar{r}_0(z) + \bar{r}_1(z), & \deg \bar{r}_{-1} = \deg q_1 + \deg \bar{r}_0\,, & \deg \bar{r}_0 > \deg \bar{r}_1 \\ \nu_2 \bar{r}_0(z) = \bar{q}_2(z)\, \bar{r}_1(z) + \bar{r}_2(z), & \deg \bar{r}_0 = \deg \bar{q}_2 + \deg \bar{r}_1\,, & \deg \bar{r}_1 > \deg \bar{r}_2 \\ \vdots & \vdots & \vdots \\ \nu_i \bar{r}_{i-2}(z) = \bar{q}_i(z)\, \bar{r}_{i-1}(z) + \bar{r}_i(z), & \deg \bar{r}_{i-2} = \deg q_i + \deg \bar{r}_{i-1}, & \deg r_{i-1} > \deg \bar{r}_i \\ \vdots & \vdots & \vdots \end{array} \qquad (17)$$

Corresponding to (5), (6), and (7), we obtain

$$\begin{pmatrix} \nu_i \bar{r}_{i-2}(z) \\ \nu_i \bar{r}_{i-2}(z) \end{pmatrix} = \begin{pmatrix} \bar{q}_i(z) & 1 \\ \nu_i & 0 \end{pmatrix} \begin{pmatrix} \bar{r}_{i-1}(z) \\ \bar{r}_i(z) \end{pmatrix},$$

$$\bar{U}_i(z) = \bar{q}_i(z)\, \bar{U}_{i-1}(z) + \nu_i \bar{U}_{i-2}(z),$$

and

$$\begin{pmatrix} \bar{U}_i(z) & \bar{U}_{i-1}(z) \\ \bar{V}_i(z) & \bar{V}_{i-1}(z) \end{pmatrix} = \begin{pmatrix} q_1(z) & 1 \\ \nu_1 & 0 \end{pmatrix} \begin{pmatrix} \bar{q}_2(z) & 1 \\ \nu_2 & 0 \end{pmatrix} \cdots \begin{pmatrix} \bar{q}_i(z) & 1 \\ \nu_i & 0 \end{pmatrix}. \qquad (18)$$

When we set $\bar{r}_{-1}(z) = r_{-1}(z)$ and $\bar{r}_0(z) = r_0(z)$, polynomials defined in this section correspond to the polynomials defined in Section 2 as shown in Table I.

TABLE I

Correspondence of Polynomials in the Method Using Euclid's Algorithm and Polynomials When Eliminating Inversion

| | $i = 2k - 1$ ($i$: odd) | $i = 2k$ ($i$: even) |
|---|---|---|
| $\bar{q}_i(z)$ | $\left(\prod_{j=1}^{k} \nu_{2j-1} \Big/ \prod_{j=1}^{k-1} \nu_{2j}\right) q_{2k-1}(z)$ | $\left(\prod_{j=1}^{k} \nu_{2j} \Big/ \prod_{j=1}^{k} \nu_{2j-1}\right) q_{2k}(z)$ |
| $\bar{r}_i(z)$ | $\left(\prod_{j=1}^{k} \nu_{2j-1}\right) r_{2k-1}(z)$ | $\left(\prod_{j=1}^{k} \nu_{2j}\right) r_{2k}(z)$ |
| $\bar{U}_i(z)$ | $\left(\prod_{j=1}^{k} \nu_{2j-1}\right) U_{2k-1}(z)$ | $\left(\prod_{j=1}^{k} \nu_{2j}\right) U_{2k}(z)$ |
| $\bar{V}_i(z)$ | $\left(\prod_{j=1}^{k} \nu_{2j-1}\right) V_{2k-1}(z)$ | $\left(\prod_{j=1}^{k} \nu_{2j}\right) V_{2k}(z)$ |

From Table I, it is seen that the corresponding polynomials are different only by scalar factors. Therefore it is evident that we can eliminate the inversion when solving the key equation for decoding Goppa codes. However, we notice for nonbinary Goppa codes that the computations of the error values does require the divisions of elements in $GF(q^m)$.

## 5. Complexity

We evaluate the complexity of our method by calculating the number of memories and the number of multiplications of elements in $GF(q^m)$ required for the inversionless algorithm.

The numbers of memories required for $\bar{r}_{i-2}(z)$, $\bar{r}_{i-1}(z)$, and $\bar{r}_i(z)$ are $(2t+1)\,m(\log_2 q)$, $2tm(\log_2 q)$, $(2t-1)\,m(\log_2 q)$, respectively, for $\bar{q}_i(z)$, $(t+1)\,m(\log_2 q)$, and for $\nu_i$, $m(\log_2 q)$. The numbers of memories required for $\bar{U}_i(z)$ and $\bar{U}_{i-1}(z)$ are $(t+1)\,m(\log_2 q)$ and $tm(\log_2 q)$, respectively. The total number of memories is approximately $(9t+3)\,m(\log_2 q)$.

The number of multiplications $N_i^{(1)}$ that are required for (17) is

$$N_i^{(1)} = 2 \deg \bar{r}_{i-2}(\deg \bar{q}_i + 1) - (\deg \bar{q}_i)^2 + 2(\deg \bar{q}_i) + 3.$$

The number of multiplications that are required for (18) is

$$N_i^{(2)} = \deg q_i \left( \sum_{j=1}^{i-1} \deg \bar{q}_i \right) + 2 \sum_{j=1}^{i-2} \deg q_j + \deg \bar{q}_{i-1} + \deg q_i + 2.$$

The total number of multiplications $N_{\text{total}}$ is

$$N_{\text{total}} = \sum_{i=1}^{k} (N_i^{(1)} + N_i^{(2)}),$$

where $k$ is the number of iterations which satisfies the relation $\sum_{i=1}^{k} \deg \bar{q}_i = e$. The values of $\deg \bar{q}_i$ and $k$ that maximize $N_{\text{total}}$ for given $t$ and $e$ are given by $\deg \bar{q}_i = 1$ and $k = e$, respectively. Therefore the maximum value of the total number of multiplications $N_{\max}$ is given by

$$N_{\max} = 8te - (e^2/2) + (13e/2).$$

For $e = t$, $N_{\max}$ is approximately $7.5t^2$.

Let us compare the complexity of our method with that of Burton's modification of Berlekamp–Massey algorithm for decoding BCH codes. Burton describes the modification of the decoding algorithm only for binary BCH codes. However, it is applicable also for nonbinary BCH codes. In the following, we use the symbols defined by Berlekamp without explanation. The number of memories for $S(z)$ is $2tm(\log_2 q)$ and those for $\sigma(z)$, $\omega(z)$, $\tau(z)$, and $\gamma(z)$ are $tm(\log_2 q)$, respectively. The total number of memories are approximately $6tm(\log_2 q)$. The number of multiplications of elements in $GF(q^m)$ depends on the case. Therefore we assume the following conditions as a typical case.

$$\deg \sigma^{(i)} = \begin{cases} [(i+1)/2] & \text{if } i \leqslant 2e, \\ e & \text{if } i > 2e, \end{cases}$$

$$\deg \omega^{(i)} = \begin{cases} [i/2] & \text{if } i \leqslant 2e, \\ e & \text{if } i > 2e, \end{cases}$$

$$\deg \tau^{(i)} = \begin{cases} [i/2] & \text{if } i \leqslant 2e, \\ i-e & \text{if } i > 2e, \end{cases}$$

$$\deg \gamma^{(i)} = \begin{cases} [(i-1)/2] & \text{if } i \leqslant 2e, \\ i-e-1 & \text{if } i > 2e, \end{cases}$$

$$\Delta_1^{(i)} \begin{cases} \neq 0 & \text{if } i \leqslant 2e, \\ = 0 & \text{if } i > 2e, \end{cases}$$

where $[x]$ is the greatest integer not greater than $x$. The total number of multiplication is given by

$$\sum_{i=1}^{2t} \{2 \deg \sigma^{(i)} + \deg \omega^{(i)} + \deg \tau^{(i)} + \deg \gamma^{(i)} + 5\}.$$

Calculating this value, the total number of multiplications is proved $4t^2 + 2te - e^2 + 10t + e$.

For $e = t$, it is approximately $5t^2$. When the multiplications can be eliminated with $\Delta_1^{(i)} = 0$, the total number of multiplications becomes approximately $2te + 3e^2$. For binary case, the number of memories is approximately $4tm$. The number of multiplications is approximately $t^2 + te - (e^2/2)$ when the multiplications cannot be eliminated with $\Delta_1^{(i)} = 0$ and is approximately $2te - (e^2/2)$ otherwise.

Therefore the number of memories required for the proposed method for decoding Goppa codes is one and a half of the number of memories required for the Burton's modification for decoding nonbinary BCH codes and is about twice for decoding binary BCH codes. The number of multiplications of elements in $GF(q^m)$ required for the proposed method is 1.5–4 times as much as the number of multiplications required for the Burton's modification for decoding nonbinary BCH codes and 4–5 times, for binary BCH codes.

## 6. Concluding Remarks

The method for solving the key equation for any Goppa polynomial $g(z)$ described in this paper requires approximately twice or thrice as much as memories and the number of multiplications compared with that of Burton's modification of Berlekamp–Massey algorithm for the special Goppa polynomial $g(z) = z^{2t}$, i.e., for BCH codes. However the method can be applied for solving the key equation for any $g(z)$ and the method may be easily understood even for nonspecialist in coding theory since the idea is simple. The property described in the corollary that the number of iterations is less than or equal to the number of errors actually occurred would be useful for error control systems with erasure options.

Professor E. R. Berlekamp has judiciously pointed out that recently two works related to our paper has been done (Berlekamp, 1974). The first (Mills, 1973) describes how the Berlekamp–Massey algorithm is related to the continued fractions algorithm. The second is an Algebraic Decoding Algorithm for Goppa Codes found by Nick Patterson (to be published in

*IEEE Trans. Inform. Theory*). It is more directly related to the Berlekamp-Massey algorithm compared with our algorithm. Computationally it seems to be quite close to our algorithm (Berlekamp, 1974).

## Acknowledgments

The authors would like to thank Professor E. R. Berlekamp very much for his warm-hearted and timely suggestions to guide us to the present subject and many helpful suggestions. The authors are also grateful to Professor T. Kasami for his warmhearted suggestions.

Received: May 31, 1974

## References

Berlekamp, E. R. (1968), "Algebraic Coding Theory," pp. 176–240, McGraw-Hill, New York.

Berlekamp, E. R. (1973), Goppa codes, *IEEE Trans. Inform. Theory* **IT-19**, 590–592.

Berlekamp, E. R. (1974), private communication.

Burton, H. O. (1971), Inversionless decoding of binary BCH codes, *IEEE Trans. Inform. Theory* **IT-17**, 464–466.

Goppa, V. D. (1970), A new class of linear error correcting codes (in Russian), *Probl. Peredach. Inform.* **6**, 24–30.

Goppa, V. D. (1971), Rational representation of codes and $(L, g)$ codes (in Russian), *Probl. Peredach. Inform.* **7**, 41–49.

Goppa, V. D. (1972), Some codes constructed on the basis of $(L, g)$ codes (in Russian), *Probl. Peredach. Inform.* **8**, 107–109.

Massey, J. L. (1969), Shift register synthesis and BCH decoding, *IEEE Trans. Inform. Theory* **IT-15**, 122–127.

Mills, W. H. (1973), "Continued Fractions and Linear Recurrences," Working Paper no. 393, Institute for Defense Analyses, Princeton, *Math. Comp.*, to be published.