J. Appl. Math. & Informatics Vol. **31***(2013), No. 5 - 6, pp. 785 - 794*
http://dx.doi.org/10.14317/jami.2013.785

A COMPLEXITY-REDUCED INTERPOLATION ALGORITHM FOR SOFT-DECISION DECODING OF REED-SOLOMON CODES[†]

KWANKYU LEE

Abstract. Soon after Lee and O'Sullivan proposed a new interpolation algorithm for algebraic soft-decision decoding of Reed-Solomon codes, there have been some attempts to apply a coordinate transformation technique to the new algorithm, with a remarkable complexity reducing effect. In this paper, a conceptually simple way of applying the transformation technique to the interpolation algorithm is proposed.

AMS Mathematics Subject Classification: 94B35, 94B27.
Key words and phrases: Reed-Solomon codes, algebraic soft-decision decoding, interpolation algorithm.

1. Introduction

Sudan and Guruswami's list decoding of Reed-Solomon codes [11, 1] developed into an algebraic soft-decision decoding by Koetter and Vardy [6]. One of the main steps of the algebraic soft-decision decoding is to construct a bivariate polynomial that interpolates given points on the plane with multiplicities, satisfying certain constraint on its weighted degree. For this interpolation step, Koetter's algorithm [5] has been most popular. Recently, Lee and O'Sullivan [9] proposed an alternative algorithm that performs the interpolation step with a comparable speed. The new algorithm consists of two steps. In the first step, it constructs a basis of a certain module of bivariate polynomials. In the second step, the basis is converted to a Gröbner basis of the module with respect to a certain weight order.

Received April 26, 2013. Revised July 5, 2013. Accepted July 8, 2013.
[†]This work was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2009-0064770) and also by research fund from Chosun University, 2008.

On the other hand, Koetter and Vardy [4] had discovered that the problem of the interpolation step can be reduced to a much smaller interpolation problem by the so-called re-encoding coordinate transformation technique. By using Koetter's algorithm to solve the reduced interpolation problem, they obtained an interpolation algorithm with a remarkably reduced complexity. Naturally, therefore, after the appearance of Lee and O'Sullivan's algorithm, there have been some attempts to apply the same technique to the new algorithm. Indeed, in personal communications, Zhang and Zhu [12] described a method to apply the transformation technique to the algorithm with some simplifying assumptions, and Ma and Vardy [10] also reported an interpolation algorithm with a reduced complexity obtained by applying a modified version of Lee and O'Sullivan's algorithm to the reduced interpolation problem.

Here we propose a conceptually simple way of applying the coordinate transformation technique to Lee and O'Sullivan's algorithm, and present a complexity-reduced interpolation algorithm and report an experimental performance result of a software implementation of the algorithm. In Section 2, we briefly review the algebraic soft-decision decoding of Reed-Solomon codes. In Section 3, we describe the original Lee and O'Sullivan's algorithm consisting of a basis construction algorithm and a Gröbner basis conversion algorithm (see [7]). In Sections 4, we note that the Gröbner basis conversion algorithm works in a general setting (see [8]). In Section 5, we present our complexity-reduced interpolation algorithm. In Section 6, an experimental result on the speed of the algorithm is reported.

2. Soft-Decision Decoding of Reed-Solomon Codes

First we define Reed-Solomon codes. Let $\mathbb{F}$ denote a field and $\mathbb{F}[x]$ the ring of polynomials in x over $\mathbb{F}$. Denote by $\mathbb{F}[x]_s$ the set of polynomials with degree $< s$, which is an s-dimensional vector space over $\mathbb{F}$. Fix n distinct points $\alpha_1, \alpha_2, \ldots, \alpha_n$ in $\mathbb{F}$. As the evaluation map $\mathrm{ev} : \mathbb{F}[x]_n \to \mathbb{F}^n$ defined by

$$f \mapsto (f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_n))$$

is an isomorphism of $\mathbb{F}$-vector spaces, the inverse map ev^{-1} exists. The Lagrange interpolation polynomials are defined by

$$h_i = \tilde{h}_i(\alpha_i)^{-1}\tilde{h}_i, \quad \text{where } \tilde{h}_i = \prod_{j=1, j\neq i}^{n} (x - \alpha_j).$$

Recall that h_1, h_2, $\ldots$, h_n form a basis of $\mathbb{F}[x]_n$. For $v = (v_1, v_2, \ldots, v_n) \in \mathbb{F}^n$, we define

$$h_v = \mathrm{ev}^{-1}(v) = \sum_{i=1}^{n} v_i h_i \in \mathbb{F}[x]_n.$$

For $k < n$, the *Reed-Solomon code* $\mathrm{RS}(n, k)$ over $\mathbb{F}$ is defined as the image of $\mathbb{F}[x]_k$ by ev. Since $\mathbb{F}[x]_k$ is a k-dimensional subspace of $\mathbb{F}[x]_n$, it follows that $\mathrm{RS}(n, k)$ is an $[n, k]$ linear code over $\mathbb{F}$.

Before we proceed, we collect definitions and notations that will be used throughout. Let $\mathbb{F}[x, y]$ be the ring of polynomials in x and y over $\mathbb{F}$. The *multiplicity* of $f \in \mathbb{F}[x, y]$ *at the origin* is defined to be the smallest m such that a monomial of total degree m occurs in the polynomial f. The *multiplicity* of f at an arbitrary point $P = (a, b)$, denoted by $\mathrm{mult}_P(f)$, is defined as the multiplicity of f_P at the origin, where $f_P = f(x + a, y + b)$. The ring $\mathbb{F}[x, y]$ is naturally identified with the ring of polynomials in y over $\mathbb{F}[x]$. For $f \in \mathbb{F}[x, y]$, the *y-degree* of f, the degree of f as a polynomial in y over $\mathbb{F}[x]$, is denoted by $y\text{-deg}(f)$. Let $f \in \mathbb{F}[x, y]$. The $(1, u)$-*weighted degree* of f is denoted by $\deg_u(f)$. That is, variables x and y are assigned with weights 1 and u, respectively, and we define $\deg_u(x^i y^j) = i + uj$, and for a polynomial f, we define $\deg_u(f)$ as the maximal $\deg_u(x^i y^j)$ for monomials $x^i y^j$ occurring in f. Finally, monomial order $>_u$ on $\mathbb{F}[x, y]$ is defined as follows. Declare $x^{i_1} y^{j_1} >_u x^{i_2} y^{j_2}$ if $\deg_u(x^{i_1} y^{j_1}) > \deg_u(x^{i_2} y^{j_2})$ or if $\deg_u(x^{i_1} y^{j_1}) = \deg_u(x^{i_2} y^{j_2})$ and $j_1 > j_2$.

Now we describe algebraic soft-decision decoding of Reed-Solomon codes. Suppose that some codeword of $\mathrm{RS}(n, k)$ was sent through a noisy channel. The output of the channel is some probabilistic information, for each location $1 \le i \le n$, of the plausibility of each symbol $\beta \in \mathbb{F}$. The multiplicity assignment step translates the information to an indexed set

$$M = \{m_{i\beta} \mid 1 \le i \le n, \beta \in \mathbb{F}\}$$

of nonnegative integers, where $m_{i\beta}$ corresponds to the point $(\alpha_i, \beta) \in \mathbb{F}^2$ of multiplicity $m_{i\beta}$. Let $M' = \{\langle(\alpha_i, \beta)\rangle m_{i\beta} \mid 1 \le i \le n, \beta \in \mathbb{F}\}$, and define

$$I_M = \{f \in \mathbb{F}[x, y] \mid \mathrm{mult}_P(f) \ge m \text{ for } \langle P\rangle m \in M'\},$$

which is an ideal of $\mathbb{F}[x, y]$. In other words, I_M is the ideal of bivariate polynomials interpolating the points with multiplicities specified by M. The task of the interpolation step is to find the minimal element of I_M with respect to the order $>_{k-1}$. Let Q be this minimal element. The final step, the root-finding step, is to find roots (polynomials in x) of Q seen as a polynomial in y over $\mathbb{F}[x]$, and outputs a list of candidates for the sent message. The rest of paper concerns the interpolation step.

3. Lee and O'Sullivan's interpolation algorithm

A basic observation is that if I is an ideal of $\mathbb{F}[x, y]$, then the minimal polynomial of I with respect to a monomial order $>$ is the smallest element of any Gröbner basis of the ideal I with respect to $>$. This is an immediate consequence of the definition of Gröbner bases. Recall that Q is the minimal polynomial of I_M with respect to $>_{k-1}$. So we can find Q by computing a Gröbner basis of I_M with respect to $>_{k-1}$. However, computing a Gröbner basis of an ideal is generally a task of high complexity. We overcome this difficulty by using Gröbner bases of modules.

Let $l \ge 0$. Define

$$\mathbb{F}[x, y]_l = \{f \in \mathbb{F}[x, y] \mid y\text{-deg}(f) \le l\},$$

and view $\mathbb{F}[x,y]_l$ as a free module over $\mathbb{F}[x]$ with a free basis $1, y, y^2, \ldots, y^l$. With this basis, we may identify $\mathbb{F}[x,y]_l$ with $\mathbb{F}[x]^{l+1}$. Note that monomials of the module $\mathbb{F}[x,y]_l$ consist of $x^i y^j$ with $i \geq 0$ and $0 \leq j \leq l$.

A monomial order $>$ on the ring $\mathbb{F}[x,y]$ naturally induces a monomial order on the module $\mathbb{F}[x,y]_l$, which is also denoted by $>$. The notions of $(1,u)$-weighted degrees and y-degrees of monomials and polynomials in $\mathbb{F}[x,y]$ carry over to $\mathbb{F}[x,y]_l$. Thus $>_u$ is a monomial order on the module $\mathbb{F}[x,y]_l$. The notion of the minimal polynomial of a submodule of $\mathbb{F}[x,y]_l$ is the same as for an ideal of $\mathbb{F}[x,y]$.

For $l \geq 0$, we define

$$I_{M,l} = I_M \cap \mathbb{F}[x,y]_l.$$

Then $I_{M,l}$ is a submodule of $\mathbb{F}[x,y]_l$. The minimal polynomial Q of I_M with respect to $>_{k-1}$ is also the minimal polynomial of $I_{M,l}$ with respect to $>_{k-1}$ if we take l large enough. Actually l can be taken to be an upper bound of the y-degree of Q. Therefore, with this choice of l, we can find Q by computing a Gröbner basis of the submodule $I_{M,l}$ of $\mathbb{F}[x,y]_l$ with respect to $>_{k-1}$. This task turns out to be much easier than that of computing a Gröbner basis of the ideal I_M.

Let $l \geq 1$. Let S be a submodule of $\mathbb{F}[x,y]_l$ over $\mathbb{F}[x]$. Suppose that S has a basis of the form

$$\begin{cases} g_0 = & a_{00} \\ g_1 = & a_{11}y + a_{10} \\ g_2 = & a_{22}y^2 + a_{21}y + a_{20} \\ \quad\vdots & \\ g_l = & a_{ll}y^l + \cdots + a_{l2}y^2 + a_{l1}y + a_{l0} \end{cases} \tag{1}$$

where $a_{ij} \in \mathbb{F}[x]$. Then the algorithm below converts the basis to a Gröbner basis.

Algorithm G (Gröbner basis conversion algorithm)**.** Let $g_i = \sum_{j=0}^{l} a_{ij} y^j$ for $0 \leq i \leq l$ during the execution of the algorithm.

G1. Set $r \leftarrow 0$.

G2. Increase r by 1. If $r \leq l$, then proceed; otherwise go to step G6.

G3. Find $s = y\text{-deg}(\text{lt}(g_r))$. If $s = r$, then go to step G2.

G4. Set $d \leftarrow \deg(a_{rs}) - \deg(a_{ss})$ and $c \leftarrow \text{lc}(a_{rs})\text{lc}(a_{ss})^{-1}$.

G5. If $d \geq 0$, then set

$$g_r \leftarrow g_r - cx^d g_s.$$

If $d < 0$, then set, storing g_s in a temporary variable,

$$g_s \leftarrow g_r, \quad g_r \leftarrow x^{-d} g_r - c g_s.$$

Go back to step G3.

G6. Output $\{g_0, g_1, \ldots, g_l\}$ and the algorithm terminates.

Now we need to find a basis of the module $I_{M,l}$ in the form (1). Recall that $M = \{m_{i\beta} \mid 1 \le i \le n, \beta \in \mathbb{F}\}$ is given. Let $M_0 = M$. We proceed inductively on $s \ge 0$. Let $s = 0$. Suppose that $M_s = \{p_{i\beta} \mid 1 \le i \le n, \beta \in \mathbb{F}\}$. For $1 \le i \le n$, let

$$p_i = \max_{\beta \in \mathbb{F}} p_{i\beta}. \tag{2}$$

Let $L = \{1 \le i \le n \mid p_i \ge 1\}$. For each $i \in L$, let β_i be such that $p_i = p_{i\beta_i}$. Define

$$g_s = \prod_{i=1}^{n} (x - \alpha_i)^{p_i} \prod_{0 \le k < s} (y - h^{(k)}), \tag{3}$$

$$h^{(s)} = \sum_{i \in L} \beta_i h_i, \text{and} \tag{4}$$

$$M_{s+1} = \{p'_{i\beta} \mid 1 \le i \le n, \beta \in \mathbb{F}\}, \tag{5}$$

where $p'_{i\beta} = p_{i\beta} - 1$ if $i \in L$ and $\beta = \beta_i$, and $p'_{i\beta} = p_{i\beta}$ if $i \notin L$ or $i \in L$ but $\beta \ne \beta_i$. We repeat the above for $0 \le s \le l$.

Note that $y - h^{(s)}$ is a curve passing through the points (α_i, β_i) with multiplicity 1 for each $i \in L$. Note also that if L is empty, then $h^{(s)} = 0$ and $g_{s+1} = yg_s$.

Proposition 3.1. *For every $l \ge 0$, we have*

$$I_{M,l} = \langle g_0, g_1, \ldots, g_l \rangle$$

as a module over $\mathbb{F}[x]$. Moreover, y-deg$(g_s) = s$ for $s \ge 0$.

Proof. It is easy to check that y-deg$(g_s) = s$ and that $g_s \in I_M$ for $s \ge 0$, so it is sufficient to show that $\{g_0, g_1, \ldots, g_l\}$ generates the module $I_{M,l}$. Let $f \in I_{M,l}$, and suppose y-deg$(f) = s$. First, for each $1 \le i \le n$, we show that $(x - \alpha_i)^{p_i}$ divides f, where p_i is as in (2). Fix i through the following paragraph.

Let $q_\beta = m_{i\beta}$ for $\beta \in \mathbb{F}$, and $f_0 = f$. Suppose y-deg$(f_0) = t$. Note that f_0 is in the ideal

$$J(q_\beta) = \{g \in \mathbb{F}[x, y] \mid \text{mult}_{(\alpha_i, \beta)}(g) \ge q_\beta, \beta \in \mathbb{F}\}.$$

Let $q = \max_{\beta \in \mathbb{F}} q_\beta$, and choose γ such that $q = q_\gamma$. Since f_0 has multiplicity q at (α_i, γ), we can write

$$f_0 = (y - \gamma) f_1 + (x - \alpha_i)^q a(x)$$

with $f_1 \in \mathbb{F}[x, y]$ and $a(x) \in \mathbb{F}[x]$. Note that y-deg$(f_1) = t - 1$ and f_1 is in the ideal $J(q'_\beta)$ where $q'_\beta = q_\beta$ if $\beta \ne \gamma$, and $q_\beta - 1$ if $\beta = \gamma$. (If $q = 0$, then we regard $q'_\beta = 0$ for all $\beta \in \mathbb{F}$.) So we can repeat this argument with setting f_0 to f_1, until $f_1 = 0$. Then we see that $(x - \alpha_i)^{p_i}$ divides f.

Now it follows that for some $b(x) \in \mathbb{F}[x]$,

$$f - b(x) \prod_{i=1}^{n} (x - \alpha_i)^{p_i} \prod_{0 \le k < s} (y - h^{(k)})$$

is in the ideal $I_{M,l}$ and has y-degree less than s. Therefore we finish the proof by induction on s. □

To summarize, Lee and O'Sullivan's interpolation algorithm goes as follows. First, given M, choose a suitable l. Second, compute a basis of $I_{M,l}$ by the procedure above. Third, convert the basis to a Gröbner basis of $I_{M,l}$ using Algorithm G. Finally, pick out the minimal element among the Gröbner basis elements. The minimal element is the bivariate polynomial Q.

4. General conversion algorithm

Algorithm G can be viewed as a special case of the following general algorithm. We consider a submodule S of $k[x]^m$. Let $e_1 < e_2 < \cdots < e_m$ denote the standard basis of $k[x]^m$. Let $u = (u_x, u_1, u_2, \ldots, u_m)$ be a given sequence of positive integers. The u-weighted degree of a monomial $x^r e_i$ is defined to be $\deg_u(x^r e_i) = u_x r + u_i$. Thus $\deg_u(ae_i) = u_x \deg(a) + u_i$ for $a \in k[x]$. A monomial order $>_u$ of $k[x]^m$ is defined by declaring $x^r e_i >_u x^s e_j$ if $\deg_u(x^r e_i) > \deg_u(x^s e_j)$ or if $i > j$ when the weighted degrees are tied.

For $f = \sum_{i=1}^{m} a_i e_i$ with $a_i \in k[x]$, define the index of f, written $\mathrm{ind}(f)$, to be the largest i such that $a_i \neq 0$. In particular, $\mathrm{ind}(x^r e_i) = i$.

Suppose $\{G_1, G_2, \ldots, G_m\}$ is a set of generators of the module S such that $\mathrm{ind}(G_i) = i$. Then the following algorithm computes a Gröbner basis of S from the given set of generators with respect to the monomial order $>_u$.

Algorithm H (General Gröbner basis conversion algorithm)**.** Let $g_i = \sum_{j=1}^{m} a_{ij} e_j$ for $1 \le i \le m$ during the execution of the algorithm. Initialize with $g_i \leftarrow G_i$.

H1. Set $r \leftarrow 1$.

H2. Increase r by 1. If $r \le m$, then proceed; otherwise go to step H6.

H3. Set $s \leftarrow \mathrm{ind}(\mathrm{lt}(g_r))$. If $s = r$, then go to step H2.

H4. Set $d \leftarrow \deg(a_{rs}) - \deg(a_{ss})$ and $c \leftarrow \mathrm{lc}(a_{rs})\mathrm{lc}(a_{ss})^{-1}$.

H5. If $d \ge 0$, then set

$$g_r \leftarrow g_r - c x^d g_s.$$

If $d < 0$, then set, storing g_s in a temporary variable,

$$g_s \leftarrow g_r, \quad g_r \leftarrow x^{-d} g_r - c g_s.$$

Go back to step H3.

H6. Output $\{g_1, \ldots, g_m\}$ and the algorithm terminates.

5. A Complexity reducing transformation

Choose k points (α_i, β) with distinct x-coordinates α_i such that these k points have largest multiplicities $m_{i\beta}$. Let $U = \{i_t \mid 1 \leq t \leq k\}$ be the set of the indices i such that the k points are (α_{i_t}, γ_t), $1 \leq t \leq k$. Let V be the set of $n-k$ indices i not contained in U. Let $\mu_t = m_{i_t\gamma_t}$ for $1 \leq t \leq k$. Let φ be the unique polynomial of degree $\leq k-1$ interpolating the selected k points (α_{i_t}, γ_t), $1 \leq t \leq k$. We introduce a notation. For a vector $a = (a_1, a_2, \ldots, a_k)$ of k integers, we define

$$\zeta^a = \prod_{t=1}^{k} (x - \alpha_{i_t})^{a_t}.$$

Proposition 5.1. *Let $f \in I_{M,l}$. Then f can be written uniquely as*

$$f = b_l \zeta^{\nu_l} (y-\varphi)^l + b_{l-1}\zeta^{\nu_{l-1}}(y-\varphi)^{l-1} + \cdots + b_0\zeta^{\nu_0} \tag{6}$$

where $b_i \in \mathbb{F}[x]$ and the vector ν_i of k integers is defined as

$$\nu_i = (\max\{\mu_t - i, 0\} \mid 1 \leq t \leq k)$$

for $0 \leq i \leq l$.

Proof. We can write uniquely

$$f = c_l(y-\varphi)^l + c_{l-1}(y-\varphi)^{l-1} + \cdots + c_0$$

for $c_i \in \mathbb{F}[x]$. Let $1 \leq t \leq k$, and $P = (\alpha_{i_t}, \gamma_t)$. Then as f_P has multiplicity μ_t at the origin, we see inductively for i from l to 0 that $x^{\max\{\mu_t - i, 0\}}$ divides c_i. The assertion now follows. □

The proposition establishes a correspondence that determines a unique vector

$$(b_l, b_{l-1}, \ldots, b_0)$$

of the module $\mathbb{F}[x]^{l+1}$ for each $f \in I_{M,l}$. It is clear that the set of the vectors in $\mathbb{F}[x]^{l+1}$ corresponding to elements of $I_{M,l}$ constitutes a submodule of $\mathbb{F}[x]^{l+1}$. Let us denote this submodule by $I'_{M,l}$. Now we observe that as φ has degree $\leq k-1$, the minimal polynomial of the module $I_{M,l}$ with respect to $>_{k-1}$ corresponds to the minimal element of $I'_{M,l}$ with respect to a monomial order $>'_{k-1}$ of $I'_{M,l}$ defined as follows. Let $\{e_l, e_{l-1}, \ldots, e_0\}$ be the standard basis of the module $\mathbb{F}[x]^{l+1}$. We assign

$$\deg(\zeta^{\nu_i}) + i(k-1)$$

as the weight of e_i, and the monomial $x^a e_i$ of $\mathbb{F}[x]^{l+1}$ has the weight a plus the weight of e_i. The monomials are ordered in the order of weights, and if tied, the monomial with larger i dominates. We denote this monomial order of $\mathbb{F}[x]^{l+1}$ by $>'_{k-1}$.

Note that the general conversion algorithm, Algorithm H, can be used to compute the minimal element of $I'_{M,l}$ with respect to $>'_{k-1}$, once we have a basis of $I'_{M,l}$. A straightforward way to obtain a basis of $I'_{M,l}$ is to write the basis elements g_s of $I_{M,l}$ in the form (6). However, to reduce computational

complexity, we need to devise a method to directly compute the basis of $I'_{M,l}$ from M. The rest of this section is devoted to this task.

Here we return to the part of Section 3 describing the construction of a basis of $I_{M,l}$, and try to translate g_s to the corresponding vector $g'_s \in \mathbb{F}[x]^{l+1}$. Suppose that the first product of (3) is written as

$$\prod_{i=1}^{n}(x-\alpha_i)^{p_i} = \prod_{i\in V}(x-\alpha_i)^{p_i}\prod_{i\in U}(x-\alpha_i)^{p_i} = \pi\zeta^p$$

where p denotes the vector $(p_{i_1}, p_{i_2}, \ldots, p_{i_k})$, and the second product of (3) is in the form

$$\prod_{0\le k<s}(y-h^{(k)}) = \sum_{i=0}^{l} c_i\zeta^{v_i}(y-\varphi)^i$$

where v_i are some vectors of k integers. As $y-h^{(s)}$ will be multiplied to the second product of (3) when s is increased by one, we want to write the new factor as

$$y-h^{(s)} = y-\varphi+\varphi-h^{(s)} = (y-\varphi)+\psi\zeta^u$$

with some vector u of k integers and some $\psi \in \mathbb{F}[x]$. Observe that for $1 \le i \le n$ with $p_i \ge 1$, the value $(\varphi - h^{(s)})(\alpha_i)$ equals

$$\begin{cases} 0, & \text{if } i = i_t \in U,\ \beta_{i_t} = \gamma_t, \\ \varphi(\alpha_i) - \beta_i = \gamma_t - \beta_i, & \text{if } i = i_t \in U,\ \beta_{i_t} \ne \gamma_t, \\ \varphi(\alpha_i) - \beta_i, & \text{if } i \in V. \end{cases}$$

So we can set $u = (u_t \mid 1 \le t \le k)$ with $u_t = 1$ if $\beta_{i_t} = \gamma_t$ and $p_{i_t} \ge 1$, and $u_t = 0$ otherwise, and ψ to a polynomial in x of smallest degree interpolating the points

$$(\alpha_i, (\varphi(\alpha_i) - \beta_i)\zeta^u(\alpha_i)^{-1})$$

for $i \in V$ with $p_i \ge 1$ and also for those $i = i_t$ with $p_i \ge 1$ and $\beta_{i_t} \ne \gamma_t$ for some $1 \le t \le k$. Thus inductively g_s can be written as

$$\begin{aligned} g_s &= \pi\zeta^p \sum_{i=0}^{l} c_i\zeta^{v_i}(y-\varphi)^i = \sum_{i=0}^{l} \pi c_i \zeta^{p+v_i}(y-\varphi)^i \\ &= \sum_{i=0}^{l} \pi c_i d_i \zeta^{\nu_i}(y-\varphi)^i \end{aligned}$$

where $d_i = \zeta^{p+v_i-\nu_i} \in \mathbb{F}[x]$. (Here note that we need to verify that $p + v_i - \nu_i$ has no negative components, though we omit a detailed verification.) Thus the element g'_s of $\mathbb{F}[x]^{l+1}$ that corresponds to g_s will be

$$g'_s = (\pi c_l d_l, \pi c_{l-1} d_{l-1}, \ldots, \pi c_0 d_0).$$

The discussion above yields the algorithm that computes a basis of $I'_{M,l}$.

Algorithm B (Basis construction algorithm)**.** The input is $M = \{m_{i\beta} \mid 1 \leq i \leq n, \beta \in \mathbb{F}\}$. The output is $\{g'_s \in \mathbb{F}[x]^{l+1} \mid 0 \leq s \leq l\}$, a basis of $I'_{M,l}$. In the following, z represents $y - \varphi$.

B1. Choose k points (α_i, β) with distinct x-coordinates α_i such that these k points have largest multiplicities $m_{i\beta}$. Let $U = \{i_t \mid 1 \leq t \leq k\}$ be the set of the indices i such that the k points are (α_{i_t}, γ_t), $1 \leq t \leq k$. Let V be the set of $n - k$ indices i not contained in U. Let $\mu_t = m_{i_t\gamma_t}$ for $1 \leq t \leq k$. Set $\nu_i \leftarrow (\max\{\mu_t - i, 0\} \mid 1 \leq t \leq k)$ for $0 \leq i \leq l$. Compute the unique polynomial φ of degree $\leq k - 1$ interpolating the selected k points, say by Newton's interpolation formula.

B2. Initialize $p_{i\beta} \leftarrow m_{i\beta}$ for $1 \leq i \leq n, \beta \in \mathbb{F}$. Set $P \leftarrow 1$ and $s \leftarrow 0$.

B3. Set $p_i \leftarrow \max_{\beta \in \mathbb{F}} p_{i\beta}$. Let β_i be such that $p_i = p_{i\beta_i}$ for i with $p_i \geq 1$.

B4. Set $p \leftarrow (p_{i_t} \mid 1 \leq t \leq k)$. Set $\pi \leftarrow \prod_{i \in V}(x - \alpha_i)^{p_i}$.

B5. Suppose $P = \sum_{i=0}^{l} c_i \zeta^{v_i} z^i$. Set $g'_s \leftarrow (\pi c_i d_i \mid 0 \leq i \leq l)$ where $d_i = \zeta^{p+v_i-\nu_i} \in \mathbb{F}[x]$.

B6. Set $u \leftarrow (u_t \mid 1 \leq t \leq k)$ where $u_t = 1$ if $p_{i_t} \geq 1$ and $\beta_{i_t} = \gamma_t$, and $u_t = 0$ otherwise.

B7. Let W be the union of $\{i \in V \mid p_i \geq 1\}$ and $\{i_t \in U \mid p_{i_t} \geq 1, \beta_{i_t} \neq \gamma_t\}$. Compute $\psi \in \mathbb{F}[x]$ interpolating $(\alpha_i, (\varphi(\alpha_i) - \beta_i)\zeta^u(\alpha_i)^{-1})$ for $i \in W$, by Newton's interpolation formula.

B8. Set $P \leftarrow P(z + \psi\zeta^u)$. For $1 \leq i \leq n, \beta \in \mathbb{F}$, set $p_{i\beta} \leftarrow p_{i\beta} - 1$ if $p_i \geq 1$ and $\beta = \beta_i$.

B9. Increase s by 1. If $s \leq l$, then go to step B3; otherwise terminate.

Note that the power product ζ^a as well as $z = y - \varphi$ are not intended to be expanded or computed during the execution of the algorithm. They serve just as symbols.

6. Performance results

We implemented the interpolation algorithm for $[255, 239, 17]$ Reed-Solomon code over $\mathbb{F}_{256}$. Some basic optimization techniques were used at several places in the implementation. We ran the algorithm for a multiplicity matrix M with the multiplicities as shown in the table below

multiplicity	7	6	5	4	3	2	1
number of points	226	16	7	6	5	6	16

counting the required numbers of finite field multiplication and division operations. This execution took 106744 multiplications plus 29542 divisions for the reduced basis construction step by Algorithm B, and 552464 multiplications and 2129 divisions for the conversion step by Algorithm H. To compute Q from Q', additional 1594280 multiplications were required. The total is 2253488 multiplications and 31671 divisions for the whole interpolation step. This contrasts with the performance of the original Lee and O'Sullivan's algorithm. For the same M, it took 14490287 multiplications for the basis construction step and 13583342

multiplications plus 2162 divisions for the conversion step by Algorithm G. So the original Lee and O'Sullivan's algorithm takes 28073629 multiplications and 2162 divisions for the whole interplolation step.

Note that for the complexity-reduced case, it will save much complexity if all the information necessary for decoding could be obtained directly from Q', without recovering Q from Q'. Some ideas in this direction have been presented by Koetter and Vardy [4], and Zhang [12].

7. Remarks

This paper deals with algorithmic aspects of Reed-Solomon codes. Concerning combinatorial aspects of error correcting codes, see [2, 3] for instance in this journal. Finally the author thanks the referees for useful comments.

References

1. Venkatesan Guruswami and Mahdu Sudan, *Improved decoding of Reed-Solomon and algebraic-geometry codes*, IEEE Trans. Inf. Theory **45** (1999), no. 6, 1757–1767.
2. Sunghyu Han, *Notes on MDS self-dual codes*, J. Appl. Math. Inform. **30** (2012), no. 5-6, 821–827.
3. Sunghyu Han and Jon-Lark Kim, *Computational results of duadic double circulant codes*, J. Appl. Math. Comput. **40** (2012), no. 1-2, 33–43.
4. R. Koetter and A. Vardy, *A complexity reducing transformation in algebraic list decoding of Reed-Solomon codes*, Proc. IEEE Symp. Information Theory Workshop (Paris, France), April 2003, pp. 10–13.
5. Ralf Koetter, *On algebraic decoding of algebraic-geometric and cyclic codes*, Ph.D. Thesis, University of Linköping, Sweden, 1996.
6. Ralf Koetter and Alexander Vardy, *Algebraic soft-decision decoding of Reed-Solomon codes*, IEEE Trans. Inf. Theory **49** (2003), no. 11, 2809–2825.
7. Kwankyu Lee and M. E. O'Sullivan, *List decoding of Reed-Solomon codes from a Gröbner basis perspective*, J. Symbolic Comput. **43** (2008), no. 9, 645–658.
8. ———, *List decoding of Hermitian codes using Gröbner bases*, Journal of Symbolic Computation (2009), no. 44, 1662–1675.
9. Kwankyu Lee and Michael E. O'Sullivan, *An interpolation algorithm using Gröbner bases for soft-decision decoding of Reed-Solomon codes*, Proc. IEEE Symp. Information Theory (Seattle, WA.), July 2006, pp. 2032–2036.
10. Jun Ma and Alexander Vardy, *A complexity reducing transformation for the Lee-O'Sullivan interpolation algorithm*, Personal communication, February 2007.
11. Madhu Sudan, *Decoding of Reed-Solomon codes beyond the error-correction bound*, J. Complexity **13** (1997), no. 1, 180–193.
12. Xinmiao Zhang and Jiangli Zhu, *Low-complexity interpolation architecture for soft-decision Reed-Solomon decoding*, Personal communication, January 2007.

Kwankyu Lee received the B.Sc., M.Sc., and Ph.D. degrees in mathematics in 1998, 2000, and 2005, respectively, from Sogang University in Korea. He is a professor at Chosun University since 2008. His research interests include algebraic coding theory, commutative algebra, and number theory.

Department of Mathematics, Chosun University, Gwangju 501-759, Korea.
e-mail: kwankyu@chosun.ac.kr