International Journal of Physics and Mathematical Sciences ISSN: 2277-2111 (Online)
An Open Access, Online International Journal Available at http://www.cibtech.org/jpms.htm
2014 Vol. 4 (1) January-March, pp. 57-61/Vinocha et al.
Research Article

A ROOT-FINDING ALGORITHM UNDER GENERALISED TRANSFORMATION (RM CODES)

***O.P. Vinocha[1], J.S. Bhullar[2] and B.S. Brar[3]**
[1]*Ferozepur College of Engineering and Technology, Ferozepur, Punjab (India)*
[2]*Department of Applied Sciences, Malout Institute of Management and Information Technology (MIMIT), Malout, Punjab (India)*
[3]*Department of Applied Sciences, Baba Farid College of Engineering and Technology, Bathinda, Punjab (India)*
**Author for Correspondence*

ABSTRACT
In this correspondence, we use an efficient root-finding algorithm under generalized transformation ($T = z^n.\psi_{k-i-1}$, n = 1, 2, 3, ….), which finds all the roots of P(T). A generalized algorithm can be used to speed up the list-decoding of RM codes, where P(T) is non-trivial polynomial in T with co-efficients in $F_q[X_1,\ldots, X_m]$ which is a ring of polynomials in m variables with co-efficients in F_q.

Keywords: *Generalised Transformation, List-Decoding, Rm Codes, Root-Finding Algorithm, Ring Of Polynomials, Graded Lexicographical Ordering.*

INTRODUCTION
Reed-Muller (RM) codes are a generalization of RS codes. Let F_q be the finite field having q elements. Let $F_q[X_1, \ldots, X_m]$ be the ring of polynomials in m variables with co-efficients in F_q. Let F_q^m be the m-dimensional vector space over F_q, where $n = q^m$. Let $P_1, \ldots, P_n$ be an enumeration of the points of F_q^m, where $n = q^m$. The q-ary RM code of order u in m variables, is denoted by $RM_q(u,m)$, and is defined as: $RM_q(u,m) = \{(f(P_1), \ldots,(f(P_n)): f \in F_q[X_1,\ldots, X_m], \deg(f) \leq u \}$. When m = 1, the code $RM_q(u,m)$ is an RS code. $RM_q(u,m)$ is an (n,k)code, where $n = q^m$, $k = {}^{u+m}C_m$.
List decoding is a decoding method, which makes possible to recover information in the presence of errors more than the general error-correction bound. A decisive step in list-decoding is to find the roots of a polynomial with co-efficients being polynomials (or rational) functions over a finite field. Then these roots are utilized to re-construct the code words which are candidates for the transmitted codeword.
Wu et al., (2005) have presented a simple and efficient algorithm, which solves the root-finding problem for list-decoding of RM codes. They have used the graded lexicographical ordering of monomials, they have modified the algorithm presented in An Algorithm for finding the roots of the polynomials over order domains by Wu (2002), and they also proved the correctness of the algorithm. One of the features of their algorithm is that it can be used to find all roots (not only those with degree ≤ a specified integer u), in space $F_q[X_1,\ldots, X_m]$, of polynomial $H(T) = h_0 + h_1T + \ldots + h_s T^s$, i. e. the algorithm can find all the linear factors of H(T).

New List-Decoding Algorithm
Let us denote the set of non-negative integers by I_{+0}. Let us denote the set of m-tuples of non-negative integers by I_{+0}^m. Clearly, every monomial: $X_1^{a_1},\ldots,X_m^{a_m}$ in $F_q[X_1,\ldots, X_m]$ uniquely corresponds to an element $(a_1,\ldots,a_m)$ in I_{+0}^m. The graded lexicographical ordering on set of m-tuples of non-negative integers I_{+0}^m, is denoted by $<_0$, and is defined as $(a_1,\ldots,a_m) <_0 (b_1,\ldots,b_m)$ if one of the following two conditions holds: (i) $\sum_{i=1}^m a_i < \sum_{i=1}^m b_i$; (ii) $\sum_{i=1}^m a_i = \sum_{i=1}^m b_i$. For example, if m = 2, then in I_{+0}^m under graded lexicographical ordering, we must have: $(0,0) <_0 (1,0) <_0 (0,1) <_0 (2,0) <_0 (1,1) <_0 (0,2) <_0$ …….This definition gives us an ordering of monomials in $F_q[X_1,\ldots, X_m]$, i.e. $X_1^{a_1},\ldots,X_m^{a_m} <_0 X_1^{b_1},\ldots,X_m^{b_m}$ iff $(a_1,\ldots,a_m) <_0 (b_1,\ldots,b_m)$.
The space $F_q[X_1,\ldots, X_m]_{\leq v}$ is a linear(vector) space over F_q and has dimension ${}^{v+m}C_m$. Let $\{\psi_0 ,\ldots, \psi_{k-1}\}$ be a basis of $F_q[X_1,\ldots, X_m]_{\leq v}$, where $\psi_0 = 1$, and for $1 \leq j \leq k-1$, ψ_j is some monomial $X_1^{a_1},\ldots,$

$X_m^{a_m}$ with $a_1 + \ldots + a_m \le v$. We suppose that $\psi_0 <_0 \psi_1 <_0 , \ldots, <_0 \psi_{k-1}$. For the basis $\{\psi_0, \ldots, \psi_{k-1}\}$, every polynomial $g \in F_q[X_1, \ldots, X_m]_{\le v}$ can be written as: $g = g_0 \psi_0 + g_1 \psi_1 + \ldots + g_{k-1} \psi_{k-1}$, where $g_0, g_1, \ldots, g_{k-1}$ are elements of F_q. We shall use the array $\{g_{k-1}, \ldots, g_1, g_0\}$of elements of F_q to represent the polynomial $g = g_0 \psi_0 + g_1 \psi_1 + \ldots + g_{k-1} \psi_{k-1}$.
We present a new algorithm, which is as follows:
Input: A non-zero polynomial: $P(T) = p_0 + p_1T + \ldots + p_s T^s$, where $p_j \in$ space $F_q[X_1, \ldots, X_m]$, and a basis$\{\psi_0, \ldots, \psi_{k-1}\}$, of space $F_q[X_1, \ldots, X_m]_{\le v}$.
Output: A list that contains all the roots of $P(T)$ in $F_q[X_1, \ldots, X_m]_{\le v}$.
Step1: Set i = 0. Set $P_0(T) = P(T)$. **Step 2:** Substitute $T = z^n.\psi_{k-i-1}$, n = 1,2,3,…into $P_i(T)$, where z denotes an undetermined element in F_q. $P_i(z^n. \psi_{k-i-1})$ is a polynomial in the variables $X_1, \ldots, X_m$. Compute the leading co-efficient of $P_i(z^n. \psi_{k-i-1})$, which is denoted by $f_i(z)$, i.e. $f_i(z) = LC(P_i(z^n. \psi_{k-i-1}))$. Clearly, $f_i(z)$ is a polynomial in z^n and hence in z with co-efficients in F_q. **Step 3:** Find all the roots of $f_i(z)$. **Step 4:** For each of the distinct roots β_{k-i-1} of the polynomial $f_i(z)$, if $i < k - 1$, set $P_{i+1}(T) = P_i(T + \beta_{k-i-1}. \psi_{k-i-1})$, set $i \leftarrow i + 1$, and return to step 2. Otherwise, go to step 5. **Step 5:** Output all arrays $[\beta_{k-1}, \ldots, \beta_1, \beta_0]$.
We discuss various examples to illustrate our list-decoding algorithm.
Illustration 1 : Let the given polynomial is:
$P(T) = T^2 - (xy + x))T$. (1)
Comparing it with $P(T) = p_0 + p_1T + \ldots + p_s T^s$, we see that: s= 2, p_0 = absent, $p_1 = xy + x$, $p_2 = 1$. Therefore, $v = \lceil max\{(deg(p_i)-deg(p_s) / (s-i): i = 0,1, \ldots, s-1\}\rceil = 2$
So, v = 2. Hence we shall try to find all roots of P(T) in $F_2[x, y]_{\le 2}$. Clearly, the space $F_2[x, y]_{\le 2}$ has a basis $\{1, x, y, x^2, xy, y^2\}$, where $1 <_0 x <_0 y <_0 x^2 <_0 xy <_0 y^2$, because every polynomial in two variable x, y of degree ≤ 2, can be written as a linear combination of members of the set $\{1, x, y, x^2, xy, y^2\}$. Let the roots of P(T) be: $G = g_0 \psi_0 + g_1 \psi_1 + \ldots + g_6 \psi_6$. We shall find co-efficients $g_0, g_1, \ldots, g_6$ of roots G recursively by using our algorithm.
Step 1: Set i = 0. Set $P_0(T) = P(T)$.
Step 2: Substitute $T = z^n.\psi_{k-i-1}$ into $P_i(T)$ i.e. substitute $T = z^n.\psi_{k-1}$ into $P_0(T)$ (since i = 0).

$$
\begin{aligned}
\text{Therefore, } P_0(T) &= P_0(z^n.\psi_{k-1}) \\
&= P_0(z^n.\psi_5) \quad (\text{since } k = 6) \\
&= P_0(z^n y^2) \quad (\text{since } \psi_5 = y^2) \\
&= P(z^n y^2) \quad (\text{since } P_0(T) = P(T)) \\
&= (z^n y^2)^2 - (xy + x)(z^n y^2) = z^{2n} y^4 - z^n xy^3 - z^n xy^2.
\end{aligned}
$$

Leading terms are: $z^{2n} y^4, - z^n xy^3$. So, LC's are: $f_0(z) = z^{2n}, - z^n$.
Step 3: Roots of $f_0(z) = z^{2n}, - z^n$ are: z = 0 i.e. $\beta_5 = 0$.
Step 4: Now $i = 0 < (k-1)(=5)$. Set $P_{i+1}(T) = P_i(T + \beta_{k-i-1}. \psi_{k-i-1})$

$$
\begin{aligned}
\text{i.e. Set } P_1(T) &= P_0(T + \beta_{k-1}. \psi_{k-1}) \quad (\text{since } i = 0) \\
&= P_0(T + \beta_5.\psi_5) \quad (\text{since } k = 6) \\
&= P_0(T + (0).y^2) \quad (\text{since } \beta_5 = 0, \psi_5 = y^2) \\
&= P_0(T).
\end{aligned}
$$

Therefore, $P_1(T) = P_0(T)$ when $\beta_5 = 0$.
We repeat the process again and again, and continuing in this way, ultimately we shall obtain:

$f_0(z) \rightarrow \beta_5 = 0 \rightarrow \beta_4 = 0 \rightarrow \beta_3 = 0 \rightarrow \beta_2 = 0 \rightarrow \beta_1 = 0 \rightarrow \beta_0 = 0$.
and $f_0(z) \rightarrow \beta_5 = 0 \rightarrow \beta_4 = 1 \rightarrow \beta_3 = 0 \rightarrow \beta_2 = 0 \rightarrow \beta_1 = 0 \rightarrow \beta_0 = 0$.

Therefore, whole of the algorithm gives us two arrays:

[0 0 0 0 0 0]
[0 1 0 0 1 0]
$(g_5\ g_4\ g_3\ g_2\ g_1\ g_0)$

Hence roots $G = g_0 \psi_0 + g_1 \psi_1 + g_2 \psi_2 + g_3 \psi_3 + g_4 \psi_4 + g_5 \psi_5 = g_0(1) + g_1(x) + g_2(y) + g_3(x^2) + g_4(xy) + g_5(y^2)$ are: $G_1 = 0$, and $G_2 = x + xy$. Put G_1 in (1), we get: $P(G_1) = (G_1)^2 - (xy + x)(G_1) = (0)^3 - (xy + x)(0) = 0$. Therefore, G_1 is a root of (1). Put G_2 in (1),we get: $P(G_2) = (G_2)^2 - (xy + x)(G_2) = (x + xy)^2 - (xy + x)(x$

+ xy) = 0. Therefore, G_1, G_2 are roots of (1).Hence [0 0 0 0 0 0] and [0 1 0 0 1 0] constitute our decoding-list.

Illustration 2: Let the given polynomial is:

$$P(T) = T^3 - (x^2y)T^2 + (y)T - x^2y^2. \quad (2)$$

We proceed as in illustration 1, and ultimately we will get:

$f_0(z) \rightarrow \beta_9 = 0 \rightarrow \beta_8 = 0 \rightarrow \beta_7 = 0 \rightarrow \beta_6 = 0 \rightarrow \beta_5 = 0 \rightarrow \beta_4 = 0 \rightarrow \beta_3 = 0 \rightarrow \beta_2 = 0 \rightarrow \beta_1 = 0 \rightarrow \beta_0 = 0.$

and $f_0(z) \rightarrow \beta_9 = 0 \rightarrow \beta_8 = 0 \rightarrow \beta_7 = 1 \rightarrow \beta_6 = 0 \rightarrow \beta_5 = 0 \rightarrow \beta_4 = 0 \rightarrow \beta_3 = 0 \rightarrow \beta_2 = 0 \rightarrow \beta_1 = 0 \rightarrow \beta_0 = 0.$

Therefore, whole of the algorithm gives us two arrays:

[0 0 0 0 0 0 0 0 0 0]
[0 0 1 0 0 0 0 0 0 0]
$(g_9\ g_8\ g_7\ g_6\ g_5\ g_4\ g_3\ g_2\ g_1\ g_0)$

Hence roots $G = g_0\,\psi_0 + g_1\,\psi_1 + + g_2\,\psi_2 + g_3\,\psi_3 + g_4\,\psi_4 + g_5\,\psi_5 + g_6\,\psi_6 + g_7\,\psi_7 + g_8\,\psi_8 + g_9\,\psi_9$ $= g_0(1) + g_1(x) + g_2(y) + g_3(x^2) + g_4(xy) + g_5(y^2) + g_6(x^3) + g_7(x^2y) + g_8(xy^2) + g_9(y^3)$ are: $G_1 = 0$, and $G_2 = x^2y$. Put G_1 in (2), we get: $P(G_1) \neq 0$. Therefore, G_1 is not a root of (2). Put G_2 in (2), we get: $P(G_2) = 0$. Therefore, G_2 is a root of (2). So, we reject G_1 and accept G_2. Hence our decoding-list will contain only one codeword, i.e. [0 0 1 0 0 0 0 0 0 0].

Illustration 3: Let the given polynomial is:

$$P(T) = T^3 - (x^2y)T^2 + (y)T\ . \quad (3)$$

Then proceeding exactly as in illustration 1, ultimately we shall get the same arrays, i. e. [0 0 0 0 0 0 0 0 0 0], [0 0 1 0 0 0 0 0 0 0] as in illustration 2. As a result, we have: $G_1 = 0$, and $G_2 = x^2y$. Putting these in (3), we see that G_1 is root of (3), and G_2 is not. So, in this case, our decoding-list will contain only one codeword, i.e. [0 0 0 0 0 0 0 0 0 0], which is all-zero codeword.

Correctness of Our Algorithm

Under Generalised Transformation $T = z^n.\psi_{k-i-1}$, n = 1,2,3,….), the correctness of our algorithm is discussed in the form of following theorems:

Theorem 1: Let $P(T) = p_0 + p_1T + \ldots\ldots + p_s T^s$ be a non-zero polynomial with co-efficients in $F_q[X_1,\ldots\ldots, X_m]$. Then our algorithm generates a list that contains all the roots of P(T) in $F_q[X_1,\ldots\ldots, X_m]_{\leq v}$.

Proof: Let $G = g_0\,\psi_0 + g_1\,\psi_1 + \ldots\ldots\ldots.. + g_{k-1}\,\psi_{k-1} \in$ space $F_q[X_1,\ldots\ldots, X_m]_{\leq v}$ (4)

be any root of $P(T) = p_0 + p_1T + \ldots\ldots + p_s T^s$ (5)

We shall prove that the co-efficients: $g_{k-1},\ldots\ldots\ldots, g_0$ of G are found recursively by the algorithm as a result of which G will be in the output of the algorithm.

Because G is a root of P(T), so, P(G) = 0. (6)

Hence $P(G) = p_0 + p_1G + \ldots\ldots + p_s G^s$

$= p_0 + p_1(g_0\,\psi_0 + g_1\,\psi_1 + \ldots + g_{k-1}\,\psi_{k-1}) + \ldots + p_s\,(g_0\,\psi_0 + g_1\,\psi_1 + \ldots + g_{k-1}\,\psi_{k-1})^s$

is the zero polynomial in $F_q[X_1,\ldots\ldots, X_m]$.

Therefore, all the co-efficients of P(G) will coincide with the zero element in F_q.

So, clearly, in particular, leading co-efficient of P(G) will be zero,

i.e. LC(P(G)) = 0. (7)

Because, $1 <_0 \psi_0 <_0 \psi_1 <_0\ ,\ldots\ldots.., <_0\ \psi_{k-1}$, Therefore,

$LC(P(G)) = LC[p_0 + p_1(g_0\,\psi_0 + g_1\,\psi_1 + \ldots + g_{k-1}\,\psi_{k-1}) + \ldots + p_s\,(g_0\,\psi_0 + g_1\,\psi_1 + \ldots + g_{k-1}\,\psi_{k-1}]$

$= LC[p_0 + p_1\,g_{k-1}\,\psi_{k-1} + \ldots + p_s\,g_{k-1}^s\,\psi_{k-1}^s]$

$= LC[P(g_{k-1}\psi_{k-1})]$ [because $P(T) = p_0 + p_1T + \ldots\ldots + p_s T^s$]

So, $LC(P(G)) = LC[P(g_{k-1}\psi_{k-1})]$

Consider: $f_0(z) = LC[P_0(z^n.\,\psi_{k-1})]$, where z^n and hence z is an undetermined element in F_q and $P_0(T) = P(T)$. Now $f_0(z) = LC[P_0(z^n.\,\psi_{k-1})]$ is a polynomial in z^n and hence in z with co-efficients in F_q and degree ≤ s. Because our assumption is that G is a root of P(T), and $G = g_0\,\psi_0 + g_1\,\psi_1 + \ldots\ldots + g_{k-1}\,\psi_{k-1}$, $P(T) = p_0 + p_1T + \ldots\ldots + p_s T^s$, $f_0(z) = LC[P_0(z^n.\,\psi_{k-1})]$, therefore, g_{k-1} is a root of $f_0(z)$. So, g_{k-1} is found in Step 3 of the algorithm.

Now, for, i = 0,1, ……., k-2, from the definition of P_{i+1} in Step 4 of the algorithm, we will have: $P_{i+1}(T) = P_i(T + g_{k-i-1}\,\psi_{k-i-1})$. This implies:

$$
\begin{aligned}
P_{i+1}(g_0\,\psi_0 + g_1\,\psi_1 + \ldots + g_{k-i-2}\,\psi_{k-i-2}) &= P_i(g_0\,\psi_0 + g_1\,\psi_1 + \ldots + g_{k-i-2}\,\psi_{k-i-2} + g_{k-i-1}\,\psi_{k-i-1}) \\
&= P_i(g_0\,\psi_0 + g_1\,\psi_1 + \ldots\ldots + g_{k-i-1}\,\psi_{k-i-1}) \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
&= P_0(g_0\,\psi_0 + g_1\,\psi_1 + \ldots\ldots + g_{k-i-1}\,\psi_{k-i-1}) \\
&= P(g_0\,\psi_0 + g_1\,\psi_1 + \ldots\ldots + g_{k-i-1}\,\psi_{k-i-1}) \\
&= P(G) \qquad \text{(using (4))} \\
&= 0 \qquad \text{(using (6))}
\end{aligned}
$$

Therefore, $P_{i+1}(g_0\,\psi_0 + g_1\,\psi_1 + \ldots\ldots + g_{k-i-2}\,\psi_{k-i-2}) \equiv 0$. (8)

It should be noted that here $P_{i+1}(T)$ has been defined from $P_i(T)$and g_{k-i-1}; and $P_i(T)$ has been defined from and $P_{i-1}(T)$and g_{k-i} ; and so on. Therefore, (8) implies:$LC(P_{i+1}(g_{k-i-2}\,\psi_{k-i-2})) \equiv 0$. This means that g_{k-i-2} is a root of $f_{i+1}(z)$, where $f_{i+1}(z) = LC[P_{i+1}(z^n.\,\psi_{k-i-2})]$. Therefore, g_{k-i-2} is found in Step 3. So, we conclude that: $g_{k-1}, g_{k-2}, \ldots\ldots g_1, g_0$ are found by our algorithm. Hence proof of the theorem is complete.

Lemma: Let $P(T) = p_0 + p_1T + \ldots + p_s T^s$ be a non-zero polynomial, where $p_j \in F_q[X_1, \ldots, X_m]$ for $j = 0,1,\ldots,s$. Let $\psi_0, \psi_1, \ldots, \psi_d$ be a basis of space $F_q[X_1, \ldots, X_m]_{\leq v}$. Suppose $\beta \in F_q$ be a root of multiplicity r, where r is a positive integer, of the polynomial equation: $f(z) = LC[P(z^n.\,\psi_d)] = 0$, where z is an undetermined element in F_q, and LC of $P(z^n.\,\psi_d)$ is determined w.r.t. the variables $X_1, \ldots, X_m$. We define $\tilde{P}(T)$ and $\tilde{f}(z)$ as: $\tilde{P}(T) = P(T + \beta.\,\psi_d)$ and $\tilde{f}(z) = LC(\tilde{P}(z^n.\,\psi_{d-1}))$. Then $\tilde{f}(z)$ is a polynomial in z^n, hence in z of degree $\leq r$.

Theorem 2: Let $P(T) = p_0 + p_1T + \ldots + p_s T^s$ be a non-zero polynomial with co-efficients in $F_q[X_1, \ldots, X_m]$. Then output of our algorithm contains at the most s elements.

Proof: We know that our algorithm finds the roots of P(T) in $F_q[X_1, \ldots, X_m]_{\leq v}$, which is a k dimensional space. We shall use the Principle of Mathematical Induction on k to prove the theorem.

We have already seen that $f_0(z)$ is a polynomial in z^n and hence in z with co-efficients in F_q and degree $\leq$ s. Therefore, $f_0(z)$ will have at the most s roots β_{k-1}, counting the multiplicities, so we shall get at the most s arrays $[\beta_{k-1}]$ of length 1.

Now let us suppose that $f_{i-1}(z)$ has $t \leq s$ roots β_{k-i}, denoted by: $\beta_{k-1}^{(1)}, \beta_{k-1}^{(2)}, \ldots\ldots, \beta_{k-1}^{(t)}$, such that every $\beta_{k-1}^{(j)}$, is a root of multiplicity r_j, where $r_1 + r_2 + \ldots + r_t \leq s$. Also we suppose that from these roots, we get at the most $t \leq s$ arrays $[\beta_{k-1}, \ldots, \beta_{k-i}]$ of length i.

From Step 4 and Step 2 of our algorithm, for each root β_{k-i}, we construct a $P_i(T)$ and the corresponding $f_i(z)$. Therefore, by Lemma, $f_i(z)$ is a polynomial in z^n and hence in z of degree $\leq r_j$. Hence $f_i(z)$ will have at the most r_j roots β_{k-i-1}. So, in total, we can get at the most: $r_1 + r_2 + \ldots + r_t \leq s$ arrays: $[\beta_{k-1}, \ldots, \beta_{k-i}, \beta_{k-i-1}]$ of length i + 1. Therefore, by Principle of Mathematical Induction, we conclude that we have at the most s arrays $[\beta_{k-1}, \beta_{k-2}, \ldots, \beta_0]$. So, size of output list of our algorithm is at the most s. Hence proof of the theorem is complete.

Theorem 3: Let $P(T) = p_0 + p_1T + \ldots + p_s T^s$, where $p_0, p_1, \ldots, p_s \in F_q[X_1, \ldots, X_m]$, $p_s \neq 0$. Let v be defined as: $v = \lceil max\{(\deg(p_i)-\deg(p_s)) / (s-i): i = 0,1,\ldots, s-1\}\rceil$. Then any root of P(T) in $F_q[X_1, \ldots, X_m]$ has degree at the most v.

Proof: Let there be a polynomial G in $F_q[X_1, \ldots, X_m]$ with:

$$\deg(G) > \lceil max\{(\deg(p_i)-\deg(p_s)) / (s-i): i = 0,1,\ldots, s-1\}\rceil = v. \tag{9}$$

Consider: $P(G) = p_0 + p_1G + \ldots + p_s G^s$ (10)

Therefore, for i = 0,1,….., s-1, we have:

$$
\begin{aligned}
\deg(p_s.G^s) - \deg(p_i.G^i) &= \deg(p_s) - \deg(p_i) + (s-i)\deg(G) \\
&> \deg(p_s) - \deg(p_i) + (\deg(p_i) - \deg(p_s)) \quad \text{(using (9))} \\
&= 0.
\end{aligned}
$$

This implies: $\deg(p_s.G^s) > \deg(p_i.G^i)$ (11)

So, (10) & (11) implies that P(G) cannot be zero polynomial. Hence G cannot be a root of P(T), where degree of G is greater than v (from (9)). So, any root of P(T) will have degree at the most v. Therefore, proof of the theorem is complete.

Conclusion

In the discussion, we have used the concept of graded lexicographical ordering. We have taken v large enough, given by the formulation: $v = \lceil max\{(\deg(p_i)-\deg(p_s) / (s-i): i = 0,1,\ldots, s-1\}\rceil$. This helps us to find the roots of the polynomial P(T), where the polynomial P(T) is computed by the list-decoder corresponding to a received vector, say, $\mathbf{y} = (y_1, y_2, \ldots, y_n)$. In our algorithm, we have used the generalized transformation: $T = z^n.\psi_{k-i-1}$. We have put the LC of polynomial $f_i(z) = P_i(z^n.\psi_{k-i-1})$ to zero to find roots of $f_i(z)$, which form the basis of arrays, which are candidates for the transmitted codewords. In case, where leading term/terms do not contain z, there we have considered next leading term/terms which contain z and have taken their LC to be utilised. In the first illustration, we have taken polynomial P(T) of second degree and our decoding-list also contains two codewords, which also satisfy the polynomial P(T). In the second and third illustrations, the polynomial P(T) is of second degree, and by our algorithm, we are able to find two arrays, but one of these satisfies the polynomial P(T) and the other not, so that our decoding-list contains one codeword, where we have expected two codewords. Therefore, we also conclude that every element of the output-list may not be a root of polynomial P(T). It may be, it may not be. Further, we conclude that the output-list contains at the most s elements, where s is the degree of the polynomial $P(T) = p_0 + p_1T + \ldots\ldots + p_s T^s$. We also conclude that any root of P(T) has degree at the most v.

REFERENCES

Guruswami V and Sudan M (1999). Improved Decoding of Reed-Solomon and Algebraic-Geometric codes. *IEEE Transactions on Information Theory* **45(**7) 1757-1767.

Pellikaan R and Wu XW (2004). List Decoding of q-ary Reed-Muller Codes. *IEEE Transactions on Information Theory* **50(**4) 679-682.

Roth R and Ruckenstein G (2000). Efficient Decoding of Reed-Solomon Codes Beyond Half the Minimum Distance. *IEEE Transactions on Information Theory* **46(**1) 246-257.

Wu XW and Siegel PH (2001). Efficient Root-Finding Algorithm With Application to List-Decoding of Algebraic-Geometric Codes. *IEEE Transactions on Information Theory* **47(**6) 2579-2587.

Wu XW (2002). An Algorithm for Finding the Roots of the Polynomials over Order Domains. *Proceedings of IEEE International Symposium on Information Theory, Lausane, Switezerland* 202.

Geil O and Pellikan R (2002). On the Structure of Order Domains. *Finite Fields Their Applications* **8** 369-396.

Wu Xin-Wen, Kuijper Margreta and Udaya Parampalli (2005). A Root-Finding Algorithm for List Decoding of Reed-Muller Codes. *IEEE Transactions on Information Theory* **51(**3) 1190-1196.