Fast Interpolation in Algebraic Soft Decision Decoding of Reed-Solomon Codes

Vera Miloslavskaya, Peter Trifonov
Saint-Petersburg State Polytechnic University, Russia,
{veram,petert}@dcn.ftk.spbstu.ru

***Abstract*— The problem of bivariate interpolation in algebraic soft-decision decoding of Reed-Solomon codes is considered. A generalization of the binary interpolation algorithm to the case of variable root multiplicity is presented.**

I. Introduction

Reed-Solomon codes are extensively used in modern communication and data recording systems. In most cases soft decision decoding is needed to achieve the tight performance requirements imposed on such systems. Kötter-Vardy algorithm enables one to solve the problem of soft decision decoding of Reed-Solomon codes with polynomial complexity [1]. However, it still remains too complex for practical implementation.

The most computationally intensive step in the Kötter-Vardy algorithm is construction of a bivariate polynomial having a number of roots of different multiplicities. This paper presents a novel reduced complexity algorithm for solving this problem. The proposed method, hereinafter referred to as "layered", is the generalization of the binary interpolation algorithm [2] to the case of roots of variable multiplicity. Furthermore, it is shown that the complexity of the proposed method can be further reduced by adjusting the root multiplicity matrix.

The paper is organized as follows. Section II introduces the necessary definitions, notation and algorithms. The novel interpolation algorithm is described in Section III. Numeric results are given in Section IV. Finally, some conclusions are drawn.

II. Background

$(n, k, n-k+1)$ Reed-Solomon code over field $\mathbb{F}$ is defined as the set of vectors $(f(x_1), \ldots, f_1(x_n))$, where $f(x) = \sum_{i=0}^{k-1} f_i x^i$, $f_i \in \mathbb{F}$ is the message polynomial, and $x_i \in \mathbb{F}$ are distinct values called code locators.

The Kötter-Vardy algorithm can be used to find all codewords satisfying the condition

$$S_M(c) > \delta_{1,k-1}(D(M)), \tag{1}$$

where M is the root multiplicity matrix, $D(M) = \sum_{i=1}^{n} \sum_{j=1}^{|\mathbb{F}|} \frac{M_{i,j}(M_{i,j}+1)}{2}$ is the cost of multiplicity matrix M, $\delta_{1,k-1}(D)$ is the minimum $(1, k-1)$-weighted degree of a bivariate polynomial having D terms, and $S_M(c) = \sum_{i=1}^{n} M_{i,c_i}$ is the score of codeword c. (w_x, w_y)-weighted degree of some polynomial is defined as the maximum over all integer numbers $iw_x + jw_y$ such that coefficient of the term with $x^i y^j$ is not equal zero.

The algorithm consists of the following steps:

1) Construction of the root multiplicity matrix M from the received sequence. This step can be implemented in many different ways, cf. [1], [3], [4], [5].
2) [Interpolation] Construction of a bivariate polynomial with the smallest possible $(1, k-1)$-weighted degree, that has roots (x_i, y_j) of multiplicity $M_{i,j}$, $i = 1, \ldots, n$, $j = 1, \ldots, |\mathbb{F}|$. This is equivalent to solving a system of D linear equations.
3) [Factorization] Finding all polynomials $f^{(i)}(x)$, such that $Q(x, f^{(i)}(x)) = 0$ and $\deg f^{(i)}(x) < k$, where $Q(x, y)$ is the polynomial obtained in the interpolation step. An efficient algorithm for solving this problem was given in [6].
4) Construction of codewords $(f^{(i)}(x_1), \ldots, f^{(i)}(x_n))$ and selection of the most probable one among them.

Interpolation is the most computationally demanding stage of this method.

Let I_M be the ideal of polynomials having roots of multiplicity not less than $M_{i,j}$ at the points $(x_i, y_j), i = 1, \ldots, n$, $j = 1, \ldots, |\mathbb{F}|$. The interpolation polynomial needed by the above algorithm is contained in its Gröbner basis constructed for the case of $(1, k-1)$-weighted degree lexicographic ordering. Gröbner basis can be obtained by applying the Buchberger algorithm [7] to some other basis of this ideal. However, its complexity turns out to be too high for practical usage.

III. Layered interpolation

This section introduces a novel bivariate interpolation algorithm for the case of the roots of variable multiplicity. The proposed method is based on the observation that the multiplicity of roots of a product of two polynomials is given by the sum of multiplicities of their roots. In terms of polynomial ideals this can be represented as $I_{M^{(1)}+M^{(2)}} = I_{M^{(1)}} I_{M^{(2)}}$, where $M^{(s)}$ are root multiplicity matrices.

A. Bitwise decomposition of root multiplicity matrix

The proposed method is based on the binary decomposition of the root multiplicity matrix elements. That is, the root

multiplicity matrix can be represented as

$$M = \sum_{h=0}^{m} 2^h M^{(h)},$$

where $m = \max_{i,j} \lfloor \log M_{i,j} \rfloor$, and $M^{(h)}$ is the h-th layer root locator matrix with entries in $\{1, 0\}$. The h-th layer L_h is defined as the set of points (x_i, y_j) for which $M^{(h)}_{i,j} = 1$. Furthermore, each matrix $M^{(h)}$ can be represented as $M^{(h)} = \sum_{t=1}^{g_h} M^{(h,t)}$, where each row of the matrix $M^{(h,t)}$ contains at most one non-zero element, and g_h is the maximal number of 1's in $M^{(h)}$ rows. This induces a decomposition of layer L_h into g_h groups, so that each group $L_{h,t}$ contains points (x_i, y_j) with distinct x_i. For each group $L_{h,t}$ one can construct the ideal $I_{M^{(h,t)}} = \{Q(x,y) | Q(x_i, y_j) = 0, (x_i, y_j) \in L_{h,t}\}$. The ideal of polynomials having roots in layer L_h, i.e.

$$I_{M^{(h)}} = \{Q(x,y) | Q(x_i, y_j) = 0, (x_i, y_j) \in L_h\},$$

can be obtained as $I_{M^{(h)}} = \prod_{t=1}^{g_h} I_{M^{(h,t)}}$. The ideals of polynomials having roots of multiplicities given by M can be obtained as

$$I_M = (\dots (I^2_{M^{(m)}} \cdot I_{M^{(m-1)}})^2 \cdot I_{M^{(m-2)}} \dots I_{M^{(1)}})^2 \cdot I_{M^{(0)}},$$

where $I^2 = I \cdot I$, $I^0 = \mathbb{F}[x, y]$ and $I \cdot \mathbb{F}[x, y] = I$.

B. Proposed algorithm

The described approach can be implemented as follows. Let us construct a sequence of matrices

$$\widehat{M}^{(h)} = 2 \cdot \widehat{M}^{(h+1)} + M^{(h)}, \tag{2}$$

where $\widehat{M}^{(m+1)} = 0$, and $m = \lfloor \log_2 \max_{i,j} M_{i,j} \rfloor$. Hence, $\widehat{M}^{(0)} = M$. This induces a sequence of ideals $I_{\widehat{M}^{(h)}} = I^2_{\widehat{M}^{(h+1)}} \cdot I_{\widehat{M}^{(h)}}$. For each ideal in this sequence one can derive a Gröbner basis, and obtain the required polynomial as the smallest element of the basis of the last ideal.

1) Multiplication of ideals: Multiplication of the ideals can be implemented using the $Merge$ algorithm given in [2], which is shown in Figure 1. This algorithm takes as input two Gröbner bases $\mathcal{P} = (P_1(x,y), \dots, P_u(x,y))$ and $\mathcal{S} = (S_1(x,y), \dots, S_v(x,y))$ of some ideals, and constructs a Gröbner basis of a subideal $J^{(0)}$ of $J = \langle P_1(x,y), \dots, P_u(x,y) \rangle \cdot \langle S_1(x,y), \dots, S_v(x,y) \rangle$. This is carried out with the aid of the function $SubidealBasis$, which will be discussed in section III-C. The algorithm proceeds with construction of a sequence of ideals (in fact, modules) $J^{(j)} = J^{(j-1)} + \langle Q_j(x,y) \rangle$, where

$$Q_j(x,y) = \left(\sum_{i=1}^{u} \alpha_{ij} P_i(x,y) \right) \left(\sum_{i=1}^{v} \beta_{ij} S_i(x,y) \right),$$

and α_{ij}, β_{ij} are independent random values uniformly distributed over $\mathbb{F}$. The polynomials $Q_j(x,y)$ are used to eliminate common roots of polynomials in $J^{(j)}$, which cannot appear in the set of common roots of polynomials in $< \mathcal{P} > \cdot < \mathcal{S} >$. It is possible to show that the sequence $J^{(j)}$ indeed converges to a Gröbner basis of J [2]. Furthermore, $J^{(j)} = J$ if and only if $\Delta(\mathcal{B}) = \Delta_0$, where $\Delta(\mathcal{B}) = \sum_{i=1}^{|\mathcal{B}|} \deg_x(B_i(x,y))$, and $\deg_x Q(x,y) = s$ provided that $\mathrm{LT}\, Q(x,y) = a x^s y^j$ [2], and $\mathrm{LT}\, Q(x,y)$ denotes the leading term of $Q(x,y)$ with respect to $(1, k-1)$-weighted degree lexicographic monomial ordering.

MERGE$((P_1(x,y), \dots, P_u(x,y)), (S_1(,y), \dots, S_v(x,y)), \Delta_0)$
1 $\mathcal{B} \leftarrow$ SUBIDEALBASIS$(\mathcal{P}, \mathcal{S})$
2 **while** $\Delta(\mathcal{B}) > \Delta_0$
3 **do** $\alpha_i \leftarrow rand(), \quad i = 1 \dots u$
4 $\beta_j \leftarrow rand(), \quad j = 1 \dots v$
5 $Q(x,y) \leftarrow ((\sum_{i=1}^{u} \alpha_i P_i(x,y))(\sum_{i=1}^{v} \beta_i S_i(x,y))$
6 $\mathcal{B} \leftarrow$ REDUCE$(\mathcal{B}, Q(,y))$
7 $\mathcal{B} \leftarrow$ MINGROEBNERBASIS$(\mathcal{B})$
8 **return** $\mathcal{B}$

Fig. 1. Construction of a Gröbner basis of $< \mathcal{P} > \cdot < \mathcal{S} >$

MINGROEBNERBASIS$(B_1(x,y), \dots, B_w(x,y))$
1 $w' \leftarrow w$
2 **while** $\deg_x(B_{w'-1}(x,y)) = 0$
3 **do** $w' \leftarrow w' - 1$
4 **if** $w' \neq w$
5 **then while** $\exists j < w' : (\deg_y(B_j(x,y)) \geq w')$
6 **do** $B_j \leftarrow B_j \mod B_{w'}$
7 **return** $(B_1(x,y), \dots, B_{w'}(x,y))$

Fig. 2. Minimization of a Gröbner basis

The algorithm makes use of the function $Reduce$, which can be considered as a multi-dimensional generalization of the extended Euclidean algorithm [2]. It takes as input a Gröbner basis $\mathcal{B}$ of some module $\mathcal{M} \subset \mathbb{F}[x, y]$, as well as a polynomial $Q(x,y)$, and constructs a Gröbner basis of the module

$$\mathcal{M}' = \{P(x,y) + a(x)Q(x,y) | P(x,y) \in \mathcal{M}, a(x) \in \mathbb{F}[x]\}.$$

After termination of the $WHILE$ loop it may happen that a few polynomials in $\mathcal{B}$ are such that $\mathrm{LT}\, B_j(x,y) = y^{j-1}, j \geq w' - 1$. Such polynomials are redundant, and should be eliminated except the smallest one. This is implemented by means of $MinGröbnerBasis$ algorithm shown in Figure 2.

Since the leading terms of these polynomials are divisible by $\mathrm{LT}\, B_{w'-1}(x,y)$, $B_j(x,y) \mod B_{w'-1}(x,y)$ does not contain any terms divisible by $y^{w'-1}$. This allows one to exclude $B_{w'+1}(x,y), \dots, B_w(x,y)$ from the obtained Gröbner basis.

It appears that if u and v are substantially different, $Merge$ algorithm may require many iterations to converge to a Gröbner basis of the ideal product. Avoiding this problem requires one to arrange the calculations in such a way that the sizes of the bases of the ideals being multiplied are approximately equal. Since the sizes of the Gröbner bases of $I_{M^{(h,t)}}$

GROUPBASIS(M)

1 $\phi(x) \leftarrow \prod_{i:M_{i,j}=1} (x - x_i)$

2 $T(x) \leftarrow \sum_{i,j:M_{i,j}=1} y_j \frac{\prod_{i':M_{i',j}=1, i' \neq i} (x - x_{i'})}{\prod_{i':M_{i',j}=1, i' \neq i} (x_i - x_{i'})}$

3 $\mathcal{B} \leftarrow (\phi(x))$

4 $h \leftarrow 0$

5 **repeat**

6 $\quad B \leftarrow$ REDUCE$(\mathcal{B}, y^h(y - T(x)))$

7 $\quad h \leftarrow h + 1$

8 **until** $\mathrm{LT}\, B_h = y^{h-1}$

9 **return** $\mathcal{B}$

Fig. 3. Construction of a Gröbner basis for $I_{M^{(h,t)}}$

increases with $D(M^{(h,t)})$, one has to sort group multiplicity matrices $M^{(h,t)}$ in the ascending order of $D(M^{(h,t)})$ while computing $I_{M^{(h)}} = \prod_{t=1}^{g_h} I_{M^{(h,t)}}$, as described in the following subsection.

2) Constructing Gröbner basis of group ideal: The algorithm for finding a Gröbner basis of the ideal $I_{M^{(h,t)}}$ is shown in Figure 3. The algorithm is based on the assumption that each row of $M^{(h,t)}$ contains at most one non-zero entry.

It was shown in [8] that the ideal of polynomials having roots in points (x_i, y_i), where x_i are some distinct values, is given by $I =< \phi(x), y - T(x) >$, where $\phi(x) = \prod_i (x - x_i)$ and $T(x_i) = y_i$. Such polynomials are constructed in lines 1–2 of this algorithm, where x_i and y_i are given by non-zero elements of group root locator matrix M. The algorithm $GroupBasis$ proceeds by constructing a sequence of modules $\mathcal{M}^{(h+1)} = \{Q(x,y) + a(x)y^h(y - T(x)) | Q(x,y) \in \mathcal{M}^{(h)}\}, h \geq 0$, where $\mathcal{M}^{(0)} = [\phi(x)]$. For each h the $Reduce$ algorithm is used to derive a Gröbner basis of this module with respect to $(1, k-1)$-weighted degree lexicographic ordering. As soon as one obtains a module with a Gröbner basis containing a polynomial $B_h(x,y) : \mathrm{LT}\, B_h(x,y) = y^{h-1}$, the $REPEAT$ loop (lines 5–8) can be terminated, since this Gröbner basis is also the one of $I =< \phi(x), y - T(x) >$. This is based on the fact that a Gröbner basis of a zero-dimensional ideal must contain a polynomial $B(x,y) : \mathrm{LT}\, B(x,y) = y^h$ for some h [9].

3) From group ideal to layer ideal: The ideal of polynomials vanishing at the points included into the h-th layer can be obtained as a product of the ideals for the corresponding groups. The algorithm presented in Figure 4 constructs a decomposition of a layer specified by its root locator matrix M into g groups. Recall, that each group consists of points (x_i, y_j) with distinct x_i, such that $M_{i,j} = 1$. The algorithm constructs first the groups corresponding to x_i most frequently used in the considered layer (they correspond to the largest c_i values). These points are excluded from further consideration.

LAYERBASIS(M)

1 $\mathcal{B} \leftarrow (1)$

2 **for** $i \leftarrow 1$ **to** n

3 **do** $c_i \leftarrow \mathrm{wt}(M_{i,-})$

4 $g \leftarrow \max_i(c_i)$

5 **for** $t \leftarrow g$ **to** 1

6 **do** $M^{(t)} \leftarrow \mathbf{0}$

7 $\quad$ **for** $i \leftarrow 1$ **to** n

8 $\quad$ **do if** $c_i \geq t$

9 $\quad\quad$ **then** Select j: $M_{i,j} = 1$

10 $\quad\quad\quad M^{(t)}_{i,j} \leftarrow 1$

11 $\quad\quad\quad M_{i,j} \leftarrow 0$

12 $\quad \mathcal{G} \leftarrow$ GROUPBASIS$(M^{(t)})$

13 $\quad \mathcal{B} \leftarrow$ MERGE$(\mathcal{B}, \mathcal{G})$

14 **return** $\mathcal{B}$

Fig. 4. Construction of a Gröbner basis for $I_{M^{(h)}}$

LAYEREDINTERPOLATION(M)

1 $m \leftarrow \max_{i,j}(\log M_{i,j})$

2 Let $M = \sum_{h=0}^{m} 2^h M^{(h)}, \quad M^{(h)}_{i,j} \in \{0, 1\}$

3 $\mathcal{B} \leftarrow$ LAYERBASIS$(M^{(m)})$

4 **for** $h \leftarrow m - 1$ **to** 0

5 **do** $\mathcal{B} \leftarrow$ MERGE$(\mathcal{B}, \mathcal{B})$

6 $\quad \mathcal{G} \leftarrow$ LAYERBASIS$(M^{(h)})$

7 $\quad \mathcal{B} \leftarrow$ MERGE$(\mathcal{B}, \mathcal{G})$

8 **return** $\mathcal{B}$

Fig. 5. Construction of a Gröbner basis for I_M

For each group the $GroupBasis$ algorithm is used to construct a Gröbner basis of the ideal of polynomials vanishing on it. $Merge$ is used to multiply the ideals corresponding to the obtained groups. Since the groups are constructed in the ascending order of their sizes, the number of polynomials in $\mathcal{B}$ and $\mathcal{G}$ do not differ significantly. This helps to avoid convergence slowdown in $Merge$.

4) Interpolation with variable root multiplicity: The proposed interpolation algorithm for the case of roots of variable multiplicity is given in Figure 5. It exploits the binary decomposition of root multiplicity matrix M given by (2) to construct a sequence of Gröbner bases for ideals $I_{\widehat{M}^{(h)}}$. Namely, for each h it invokes $Merge$ to compute $I^2_{\widehat{M}^{(h+1)}}$ (line 5), constructs a Gröbner basis of the ideal of polynomials vanishing on the h-th layer (line 6), and obtains $I_{\widehat{M}^{(h)}}$ after a call to $Merge$ on line 7.

The interpolation polynomial needed by the Kötter-Vardy soft-decision decoding algorithm is given by the smallest element of basis $\mathcal{B}$ constructed by this algorithm [10].

C. Improving the convergence speed of $Merge$

The convergence speed of the $Merge$ algorithm shown in Figure 1 depends on the selection of the initial basis (see line

1). The simplest choice is given by [2]

$$B_i(x,y) = P_{i-j+1}(x,y)S_j(x,y), 1 \le i \le u+v-1, \quad (3)$$

where j is selected in such way that LT $B_i(x,y)$ is minimized. This allows one to improve the performance of the $Reduce$ algorithm. However, if the sizes of the Gröbner bases of the ideals being multiplied are significantly different, this method causes some polynomials to be used too frequently. The ideal generated by polynomials $B_i(x,y)$ constructed in this way may have too many extra zeroes. Since the objective of lines 2–6 of $Merge$ is elimination of these unwanted zeroes, such poor selection of the initial basis may require a lot of iterations before the algorithm converges to a Gröbner basis of $<\mathcal{P}>\cdot<\mathcal{S}>$.

Hence, we suggest to impose a limit on the maximal number of times a polynomial can be used while constructing the initial basis. Namely, let $s_{i,j} = \begin{cases} 1, & S_j(x,y) \text{ is used in (3)} \\ 0, & \text{otherwise} \end{cases}$. We propose to minimize

$$\Delta(\mathcal{B}) = \sum_i \sum_j s_{i,j} \left(\deg_x P_{i-j+1}(x,y) + \deg_x S_j(x,y)\right)$$

under the constraints

$$\sum_i s_{i,j} \le U_0 \quad (4)$$

$$\sum_i s_{i,i-j+1} \le U_1 \quad (5)$$

$$\sum_j s_{i,j} = 1, \quad (6)$$

where the parameters U_0 and U_1 should be selected in such way that the polynomials in $\mathcal{P}$ and $\mathcal{S}$ are uniformly used for initialization of $Merge$.

This is an instance of boolean programming problem, and the search for its optimal solution requires significant computational effort. Hence, we propose the following suboptimal approach. One can greedily select the smallest $\mathcal{B}_i(x,y)$ in (3) until a limit on the number of uses of each polynomial is exceeded. In some cases it may be impossible to satisfy this constraint. In this case an arbitrary pair of polynomials is selected. The proposed algorithm is summarized in Figure 6.

D. Adjusting the root multiplicity matrix

The most computationally expensive step of layered interpolation algorithm is multiplication of ideals of polynomials having roots with large multiplicity by ideals of polynomials having many roots of multiplicity one. So, we propose to subtract $M^{(0)}$ from root multiplicity matrix, i.e. exclude layer L_0. Numeric results presented below indicate that this results in a negligible performance degradation, while significantly decreasing the computational complexity of the interpolation algorithm.

SUBIDEALBASIS$((\mathcal{P},\mathcal{S}))$

```
1  π ← [0, ..., 0]
2  σ ← [0, ..., 0]
3  μ ← (u+v−1)^2 / (u·v)
4  for i ← 1 to u + v − 1
5  do j_0 ← i − min(i, u) + 1
6     d_0 ← deg_x P_{i−j_0+1}(x,y) S_{j_0}(x,y)
7     for j ← 1 to u
8     do if deg_x P_{i−j+1}(x,y) S_j(x,y) < d_0
9           then if π_{i−j+1} + σ_j < μ
10                 then j_0 ← j
11                      d_0 ← deg_x P_{i−j_0+1}(x,y) S_{j_0}(x,y)
12                      π_{i−j+1} ← π_{i−j+1} + 1
13                      σ_j ← σ_j + 1
14    B_i ← P_{i−j_0+1}(x,y) S_{j_0}(x,y)
15 return B
```

Fig. 6. Construction of a Gröbner basis for an initial subideal of $\mathcal{P}\cdot\mathcal{S}$

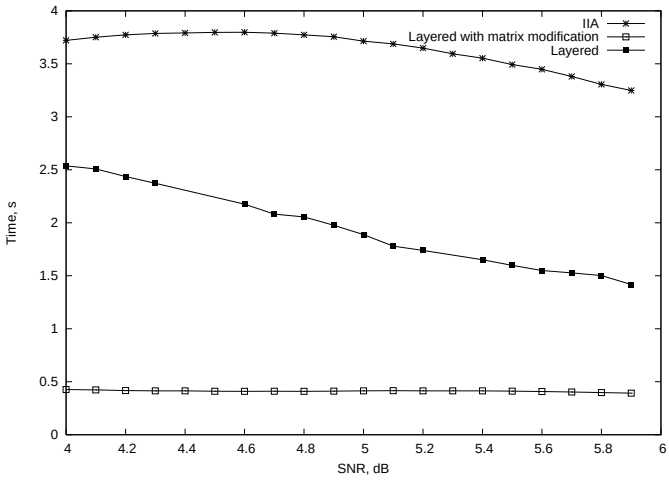

Fig. 7. Decoding complexity with IIA and proposed algorithm

IV. NUMERIC RESULTS

The proposed algorithms were implemented in C++ programming language, and computer simulations were used for analysis of their complexity and performance. Transmission of the binary image of Reed-Solomon codes over the AWGN channel was considered.

Figure 7 illustrates the decoding time of $(63, 48)$ code over $GF(2^6)$ with the Kötter-Vardy method based on the iterative interpolation algorithm [11] and the proposed one. It can be seen that the layered interpolation algorithm provides more than two times complexity reduction, and the gain increases with SNR. The reason for this is that for higher SNR the layers consist of smaller number of groups. This results in smaller number of calls to $Merge$ on line 13 of the $LayerBasis$ algorithm. It can be also seen that eliminating the 0-th layer from the multiplicity matrix leads to more than five time complexity reduction at low SNR.

Figure 8 presents performance comparison of soft-decision

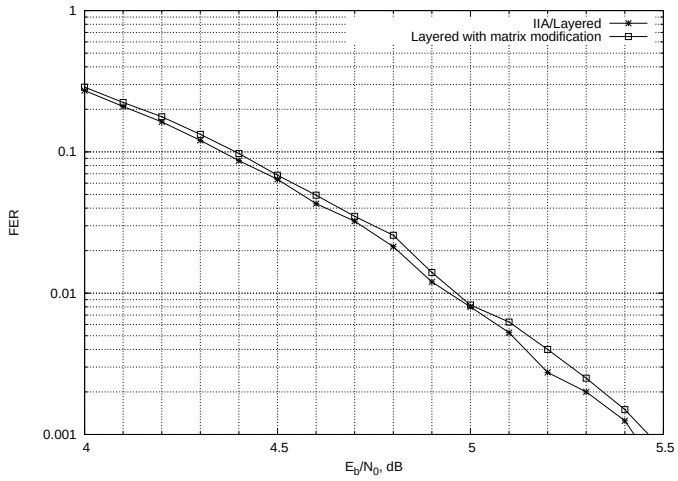

Fig. 8. Decoding performance with original and modified matrices

decoding algorithms utilizing the original root multiplicity matrix obtained according to the description given in [1], and the one without the 0-th layer. It can be seen that the proposed modification results in approximately 0.05 dB performance degradation. Observe that the layered interpolation algorithm by itself (without multiplicity matrix modification) does not affect the performance of the algebraic soft decision decoder.

It appears that the most computationally intensive part of the proposed method is the $Reduce$ algorithm. Any improvement in its complexity would automatically result in complexity reduction of the layered interpolation algorithm.

V. Conclusion

In this paper a novel bivariate interpolation algorithm for algebraic soft-decision decoding of Reed-Solomon codes was proposed. This algorithm is a generalization of the binary interpolation algorithm presented in [2] to the case of variable root multiplicity. Numeric results indicate that it provides more than two times complexity reduction compared to the case of iterative interpolation algorithm.

Furthermore, it was shown that additional three times complexity reduction can be achieved by introducing minor modifications to root multiplicity matrix at the expense of negligible performance degradation.

The complexity of the proposed method can be further reduced by integrating it with the re-encoding transformation [12], and developing more efficient alternatives to the multi-dimensional Euclidean algorithm ($Reduce$).

References

[1] R. Koetter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes," *IEEE Transactions on Information Theory*, vol. 49, no. 11, pp. 2809–2825, November 2003.

[2] P. Trifonov, "Efficient interpolation in the Guruswami-Sudan algorithm," *IEEE Transactions on Information Theory, accepted for publication*, 2010.

[3] J. Jiang and K. R. Narayanan, "Algebraic soft-decision decoding of reed-solomon codes using bit-level soft information," *IEEE Transactions On Information Theory*, vol. 54, no. 9, September 2008.

[4] F. Parvaresh and A. Vardy, "Multiplicity assignments for algebraic soft-decoding of Reed-Solomon codes," in *Proceedings of IEEE International Symposium on Information Theory*, June 2003, p. 205.

[5] H. Das and A. Vardy, "Multiplicity assignments for algebraic soft-decoding of reed-solomon codes using the method of types," in *Proceedings of IEEE International Symposium on Information Theory*, 2009.

[6] R. Roth and G. Ruckenstein, "Efficient decoding of Reed-Solomon codes beyond half the minimum distance," *IEEE Transactions on Information Theory*, vol. 46, no. 1, pp. 246–257, 2000.

[7] D. Cox, G. Little, and D. O'Shea, *Ideals, varieties and algorithms*. Springer-Verlag, 1992.

[8] K. Lee and M. E. O'Sullivan, "List decoding of Reed-Solomon codes from a Gröbner basis perspective," *Journal of Symbolic Computation*, vol. 43, no. 9, September 2008.

[9] T. Becker and V. Weispfenning, *Gröbner Bases. A Computational Approach to Commutative Algebra*. New York: Springer, 1993.

[10] T. Sauer, "Polynomial interpolation of minimal degree and Gröbner bases," in *Gröbner Bases and Applications (Proceedings of the International Conference "33 Years of Gröbner Bases")*, ser. London Mathematical Society Lecture Notes, B. Buchberger and F. Winkler, Eds., vol. 251. Cambridge University Press, 1998, pp. 483–494.

[11] R. R. Nielsen and T. Hoholdt, "Decoding Reed-Solomon codes beyond half the minimum distance," in *Proceedings of the International Conference on Coding Theory and Cryptography*. Mexico: Springer-Verlag, 1998.

[12] R. Koetter, J. Ma, A. Vardy, and A. Ahmed, "Efficient interpolation and factorization in algebraic soft-decision decoding of Reed-Solomon codes," in *Proceedings of IEEE International Symposium on Information Theory*, Yokohama, Japan, June 29 – July 4 2003, p. 365.