Hybrid Interpolation Algorithm for 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. Re-encoding transformation for the case of layered interpolation algorithm is derived. A novel hybrid interpolation algorithm based on layered, Lee-O'Sullivan and Kötter algorithms is proposed. The proposed algorithm demonstrates considerable complexity reduction compared to the case of iterative interpolation algorithm.**

I. Introduction

Interpolation is known to be the most computationally demanding step in the Kötter-Vardy algorithm [1]. In this paper we propose a novel reduced complexity hybrid method for solving this problem. The proposed approach is based on the layered algorithm [2] integrated with re-encoding transformation [3]. Some parts of the interpolation problem are solved with Lee-O'Sullivan [4] and Kötter methods [5], [6].

The paper is organized as follows. Section II introduces the necessary definitions, notation and algorithms. Re-encoding transformation for the case of layered interpolation algorithm is described in Section III. The novel interpolation method is presented in Section IV. Numeric results are given in Section VI. Finally, some conclusions are drawn.

II. Background

$(n,k,n-k+1)$ Reed-Solomon code over field $\mathbb{F}$ is defined as a set of vectors $(f(x_0),\dots,f(x_{n-1}))$, 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 some distinct values called code locators.

A. Kötter-Vardy algorithm

The Kötter-Vardy algorithm involves the following steps [1]:

1) Construction of the root multiplicity matrix M from the received noisy sequence. This step can be implemented in many different ways, cf. [1], [7], [8], [9].
2) [Interpolation] Construction of a bivariate polynomial $V(x,y)$ with the smallest possible $(1,k-1)$-weighted degree, having roots (x_i,y_j) of multiplicity $M_{i,j}$, $i=0,\dots,n-1$, $j=0,\dots,|\mathbb{F}|-1$. That is, all its partial Hasse derivatives of total order less than $M_{i,j}$ at points (x_i,y_j) should be equal to zero.
3) [Factorization] Finding all polynomials $f^{(i)}(x)$, such that $V(x,f^{(i)}(x))=0$ and $\deg f^{(i)}(x)<k$. An efficient algorithm for solving this problem was given in [10].
4) Construction of codewords $(f^{(i)}(x_0),\dots,f^{(i)}(x_{n-1}))$ 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 $(x_i,y_j), i=0,\dots,n-1, j=0,\dots,|\mathbb{F}|-1$. Let us introduce the $(1,t)$-weighted degree lexicographic term ordering as $x^a y^b \prec_t x^A y^B \Leftrightarrow (a+bt<A+Bt)\vee((a+bt=A+Bt)\wedge(b<B))$. Given a polynomial $Q(x,y)$: $\operatorname{LT} Q(x,y)=ax^by^c$, where LT is the leading term of $Q(x,y)$ with respect to $\prec_{k-1}$, we define $\operatorname{xdeg} Q(x,y)=b$ and $\operatorname{ydeg} Q(x,y)=c$. The desired polynomial $V(x,y)$ can be found in a Gröbner basis of I_M constructed with respect to $\prec_{k-1}$ [11]. It is possible to show that any such Gröbner basis contains a polynomial $Q(x,y)$: $\operatorname{LT} Q(x,y)=y^\rho$ [12]. The standard technique used in the design of fast bivariate interpolation algorithms is to consider only the module $I_{M,\rho}=\{Q(x,y)=\sum_{t=0}^{\rho} q_t(x)y^t | Q(x,y)\in I_M\}$, and construct its Gröbnber basis, which is also a Gröbner basis of I_M. Given some polynomials $P_t(x,y)=\sum_{s=0}^{\rho} p_{t,s}(x)y^s$, let $[P_0(x,y),\dots,P_\rho(x,y)]=\{Q(x,y)=\sum_{t=0}^{\rho} a_t(x)P_t(x,y)|a_t(x)\in\mathbb{F}[x]\}$ be the module generated by them.

B. Lee-O'Sullivan algorithm

The algorithm presented in [4] solves the interpolation problem by constructing a basis of $I_{M,\rho}$, and transforming it into a Gröbner one with a multidimensional generalization of the Euclidean algorithm. It proceeds by decomposing the root multiplicity matrix as $M=\sum_{s=0}^{\theta} M^{(s)}$, where $M^{(s)}$ are binary matrices with at most one non-zero element in each row. For each $M^{(s)}$ we define a set of points G_s as the set $\left\{(x_i,y_j)\in\mathbb{F}^2 \,|M_{i,j}^{(s)}=1\right\}$. Observe that each set G_s contains points (x_i,y_j) with distinct x_i. For arbitrary set A of points (x_i,y_j) we define $\{A\}_x=\{i|(x_i,y_j)\in A\}$. The Lee-O'Sullivan algorithm proceeds with construction of a Gröbner basis of I_M as follows.

1) Let $\mathcal{B}=()$, $s=0$.
2) Compute $u_s(x)=\prod_{i\in\{G_s\}_x}(x-x_i)^{m_i}$, where $m_i=\max\limits_{0\le j\le|\mathbb{F}|-1}\sum\limits_{t=s}^{\theta} M_{i,j}^{(t)}$, and an interpolation polynomial $w_s(x)$, such that $w_s(x_i)=y_j$ for all $(x_i,y_j)\in G_s$.

Compute

$$g_s(x,y)=u_s(x)\prod_{t=0}^{s-1}(y-w_t(x)). \quad (1)$$

3) Construct a Gröbner basis $\widehat{\mathcal{B}}$ of the module $\{Q(x,y)+a(x)g_s(x,y)|Q(x,y)\in[\mathcal{B}]\}$ using the multidimentional Euclidean algorithm (see *Reduce* in [13]). Let $\mathcal{B}=\widehat{\mathcal{B}}$. Assume without loss of generality that $\operatorname{ydeg} B_i(x,y)=i$.
4) If $s>\theta$ and $\operatorname{xdeg} B_s(x,y)=0$, then $\mathcal{B}$ is Gröbner basis of I_M. Otherwise, let $s=s+1$ and go to step 2.

C. Iterative interpolation algorithm (IIA)

Given a Gröbner basis $\mathcal{B}$ of a module $\mathcal{N}$, the Kötter interpolation algorithm [5], [6] transforms it into a Gröbner basis of another module $\mathcal{N}'\subset\mathcal{N}$, such that all $Q(x,y)\in\mathcal{N}'$ have a additional common root (α,β) of multiplicity at least m. This is implemented by performing the following steps for all non-negative $j_1,j_2:j_1+j_2<m$:

1) For each polynomial $B_i(x,y)$ compute its Hasse derivative σ_i of order (j_1,j_2) at point (α,β).
2) Let i_0 be the index of the smallest polymomial $B_i(x,y)$ with non-zero σ_i
3) Let $B_i(x,y)=B_i(x,y)-\frac{\sigma_i}{\sigma_{i_0}}B_{i_0}(x,y)$ for all $i\neq i_0$.
4) Let $B_{i_0}(x,y)=B_{i_0}(x,y)(x-\alpha)$.

D. Layered interpolation algorithm

Layered interpolation algorithm described in [2] is based on the binary decomposition of the root multiplicity matrix elements. The root multiplicity matrix can be represented as

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

where $m=\max_{i,j}\lfloor\log M_{i,j}\rfloor$, and $M^{(h)}$ are binary matrices. We define the h-th layer of interpolation points as $L_h=\left\{(x_i,y_j)\in\mathbb{F}^2\,|M_{i,j}^{(h)}=1\right\}$.

Furthermore, each matrix $M^{(h)}$ can be represented as $M^{(h)}=\sum_{s=0}^{\theta_h}M^{(h,s)}$, where each row of the matrix $M^{(h,s)}$ contains at most one non-zero element. This induces a decomposition of layer L_h into sets G_s, $s=0,\ldots,\theta_h$. For each set G_s one can construct the ideal $I_{M^{(h,s)}}=\{Q(x,y)|Q(x_i,y_j)=0,(x_i,y_j)\in G_s\}$. 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

$$I_{M^{(h)}}=\prod_{s=0}^{\theta_h}I_{M^{(h,s)}}. \quad (2)$$

The ideal of polynomials having roots of multiplicities given by M can be obtained as $I_M=J_m$, where

$$J_{h+1}=K_hI_{M^{(m-h-1)}},K_h=J_h^2,J_0=I_{M^{(m)}}, \quad (3)$$

and $I^2=I\cdot I$. It 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. Let ρ_h be the smallest integer such that a Gröbner basis of J_{h,ρ_h} is also the one for J_h. Observe that $J_{h,\rho_h}=I_{W^{(h)},\rho_h}$, where $W^{(h)}=\sum_{t=0}^{h}2^tM^{(m-t)}$.

An efficient algorithm for construction of a Gröbner basis of a product of two zero-dimensional ideals I_{M_1} and I_{M_2} was presented in [13]. This algorithm takes as input two Gröbner bases $(P_0(x,y),\ldots,P_{\rho_1}(x,y))$ and $(S_0(x,y),\ldots,S_{\rho_2}(x,y))$ of modules I_{M_1,ρ_1} and I_{M_2,ρ_2}, such that these bases are also Gröbner bases of I_{M_1} and I_{M_2}, and employs the multidimensional Euclidean algorithm to construct a sequence of Gröbner bases $\mathcal{B}^{(t+1)},t=0,1,\ldots,$ of modules

$$\mathcal{N}^{(t+1)}=\left\{Q(x,y)+a(x)\tilde{Q}_t(x,y)|Q(x,y)\in\mathcal{N}^{(t)}\right\}, \quad (4)$$

where $\tilde{Q}_t(x,y)$ are randomly chosen polynomials in $I_{M_1}I_{M_2}$, and $\mathcal{N}^{(0)}$ is a module with a Gröbner basis given by

$$Q_j(x,y)=P_{j-i_j}(x,y)S_{i_j}(x,y),j=0,\ldots,\rho_1+\rho_2, \quad (5)$$

for some i_j. This sequence converges rapidly to a Gröbner basis of $I_{M_1+M_2}=I_{M_1}I_{M_2}$. The convergence criterion is given by

$$\Delta(\mathcal{B}^{(t)})=\Delta_{M'}, \quad (6)$$

where $M'=M_1+M_2$, and

$$\Delta_{M'}=\sum_{i=0}^{n-1}\sum_{j=0}^{|\mathbb{F}|-1}\frac{M'_{i,j}(M'_{i,j}+1)}{2},\Delta(\mathcal{B})=\sum_{i=0}^{|\mathcal{B}|-1}\operatorname{xdeg} B_i(x,y).$$

By abuse of notation, we will denote this operation as multiplication of modules I_{M_1,ρ_1} and I_{M_2,ρ_2}.

E. Re-encoding

The re-encoding transformation [3] is based on the change of variables

$$z=\frac{y-f(x)}{\phi(x)}, \quad (7)$$

where $\phi(x)=\prod_{i\in\{R\}_x}(x-x_i)$, and $f(x)$ is the smallest degree polynomial s.t. $f(x_i)=y_j$ for all $(x_i,y_j)\in R$. The set of re-encoding locators R consist of k points (x_i,y_j) with distinct x_i and highest possible $M_{i,j}$.

This transformation enables one to eliminate a large fixed factor of the interpolation polynomial $V(x,y)$, drastically reducing thus the dimension of the problem. It was initially presented in the context of IIA, and later extended to the case of Lee-O'Sullivan algorithm [14]. IIA-based implementation of the re-encoding requires one to use the modified interpolation points (x_i,z_j), where

$$z_j=\begin{cases}(y_j-f(x_i))/\phi(x_i), & i\notin\{R\}_x,\\(y_j-f(x_i))/\phi'(x_i), & i\in\{R\}_x,(x_i,y_j)\notin R,\end{cases} \quad (8)$$

and a modified expression for Hasse derivatives for the second case. For details, see [3]. One of contributions of this paper is a generalization of this approach to the case of layered interpolation algorithm.

III. Re-encoding transformation for layered interpolation algorithm

Consider construction of $I_{M^{(h)}}$. Assume that the h-th layer is decomposed into sets $G_s, s=0,\ldots,\theta_h$. The basis of $I_{M^{(h)}}$ is given by (1). After change of variables (7), it becomes

$$\widehat{B}_s(x,z)=u_s(x)\prod_{t=0}^{s-1}(\phi(x)z+f(x)-w_t(x)), s=0,\ldots,\rho_h. \tag{9}$$

Assume that G_0 contains all re-encoding locators of the h-th layer. This implies that $f(x)-w_0(x)$ is divisible by $\psi^{(h)}(x)=\prod_{i\in\{R\cap L_h\}_x}(x-x_i)$. Observe that both $u_0(x)$ and $\phi(x)$ are divisible by $\psi^{(h)}(x)$. Therefore, $\psi^{(h)}(x)$ is a common factor of all $\widehat{B}_s(x,z)$.

From (3) it follows that all polynomials in J_{h,ρ_h} are divisible by $\Psi^{(h)}(x)$, where

$$\Psi^{(h)}(x)=\prod_{t=0}^{h}(\psi^{(m-t)}(x))^{2^t}=\prod_{(x_i,y_j)\in R}(x-x_i)^{W^{(h)}_{i,j}}.$$

Eliminating this common factor one obtains another module $\widehat{J}_{h,\rho_h}$. Observe that $\widehat{J}_{m,\rho_m}$ corresponds to the same instance of the interpolation problem as $I_{M,\rho}$.

Let $\mathrm{LT}\,Q(x,y)=ax^by^c$ for some $Q(x,y)\in J_{h,\rho_h}$. After the above described transformations and switching to $\prec_{-1}$, one obtains

$$\widehat{Q}(x,z)=\frac{Q(x,\phi(x)z+f(x))}{\Psi^{(h)}(x)},$$

such that $\mathrm{LT}\,\widehat{Q}(x,z)=\widehat{a}x^{ck+b-\tau_h}z^c$, where

$$\tau_h=\deg\Psi^{(h)}(x)=\sum_{(x_i,y_j)\in R}\sum_{t=0}^{h}2^tM^{(m-t)}_{i,j}.$$

This implies that for a Gröbner basis $\widehat{\mathcal{B}}^{(h)}$ of $\widehat{J}_{h,\rho_h}$ $\widehat{\Delta}_{W^{(h)}}=\Delta(\widehat{\mathcal{B}}^{(h)})=\sum_{t=0}^{\rho_h}\mathrm{xdeg}\,\widehat{B}^{(h)}_t(x,y)=\sum_{t=0}^{\rho_h}(\mathrm{xdeg}\,B^{(h)}_t(x,y)+tk-\tau_h)=\Delta_{W^{(h)}}+k\rho_h(\rho_h+1)/2-(\rho_h+1)\tau_h$.

Hence, one can apply the re-encoding transformation while constructing the Gröbner bases for each layer L_h, and proceed according to (3), replacing the convergence criterion (6) with the above expression.

IV. Hybrid interpolation algorithm

It turns out that a straightforward implementation of the ideal construction method given by (3) is extremely inefficient if some of $M^{(h)}$ contain just a few 1's. Therefore we propose to combine the layered interpolation algorithm with other approaches for construction of J_{h+1} out of J_h.

A. *Convergence of the ideal multiplication algorithm*

In what follows, we show that the complexity of the randomized ideal multiplication algorithm given by (4) substantially depends on the properties of ideals being multiplied. The complexity of construction of a Gröbner basis of $I_{M_1}I_{M_2}$ using this approach depends on the length of the sequence of submodules $\mathcal{N}^{(0)}\supset\mathcal{N}^{(1)}\supset\ldots$ having Gröbner bases $\mathcal{B}^{(0)},\mathcal{B}^{(1)},\ldots$, such that $\Delta(\mathcal{B}^{(i)})>\Delta_{M_1+M_2}$. It increases with $N=\Delta(\mathcal{B}^{(i)})-\Delta_{M_1+M_2}$. The number of operations needed by the multidimentional Euclidean algorithm also increases with N.

For the sake of simplicity, consider the case of fixed root multiplicity (the case of list decoding). It was shown in [13] that in the case of $M_i=r_iH$, $i=1,2$, where $r_i\in\mathbb{N}$, H is a $n\times|\mathbb{F}|$ matrix containing exactly one 1 in each row

$$N\approx\frac{n}{2}\left(\frac{\rho_2}{\rho_1+1}r_1(r_1+1)+\frac{\rho_1}{\rho_2+1}r_2(r_2+1)-2r_1r_2\right),$$

where ρ_i are maximal y-degrees of polynomials in the bases of the ideals being multiplied. It was shown in [6] that

$$\rho_i(\rho_i+1)\le nr_i(r_i+1)/(k-1)<(\rho_i+1)(\rho_i+2),$$

i.e. $\rho_i+1\approx\alpha r_i$ for some α. Hence,

$$N\approx\frac{n}{2\alpha}\left((\alpha-1)(r_1+r_2)-2\right).$$

In the case of $r_1=r_2=r/2$ it becomes

$$N\approx\frac{\nu}{\alpha}\left((\alpha-1)r-1\right). \tag{10}$$

Similar value is obtained for $r_1=r, r_2=1$. This implies that the complexity of ideal multiplication is approximately the same in both cases. In the case of variable root multiplicity the complexity of computing Gröbner bases of $K_h=J_h^2$ and $J_{h+1}=K_hI_{M^{(m-h-1)}}$ is also approximately the same. Most of the matrices $M^{(h)}$ are sparse, so the number of operations needed to construct J_{h+1} from K_h via ideal multiplication is inproportionally high.

Consider the case of coprime ideals I_{M_1} and I_{M_2}, where $M_i=rH_i$, and $H=H_1+H_2$ contains exactly one 1 in each row. For the sake of simplicity assume that $\rho_1=\rho_2=\rho$, and matrices M_1, M_2 have ν nonzero elements. Hence, $\Delta_{M_1}=\Delta_{M_2}=\nu r(r+1)/2$ and $\Delta_{M_1+M_2}=\nu r(r+1)$. Let $(P_0(x,y),\ldots,P_\rho(x,y))$ and $(S_0(x,y),\ldots,S_\rho(x,y))$ be Gröbner bases of I_{M_1} and I_{M_2}, such that $\mathrm{LT}\,P_i(x,y)=\beta_ix^{p_i}y^i$, $\mathrm{LT}\,S_i(x,y)=\gamma_ix^{s_i}y^i$. It was shown in [13] that $p_i\approx s_i\approx l_r-j(k-1)$, where $l_r\approx((k-1)\rho(\rho+1)+\nu r(r+1))/(2(\rho+1))$. Assuming that the Gröbner basis $\mathcal{B}^{(0)}$ of $\mathcal{N}^{(0)}$ is given by (5), one obtains $\Delta(\mathcal{B}^{(0)})=\sum_{j=0}^{2\rho}\mathrm{xdeg}\,Q_j(x,y)\approx\frac{\nu}{\alpha}(r+1)(2\alpha r-1)$. Hence,

$$N\approx\frac{\nu}{\alpha}(r+1)(\alpha r-1). \tag{11}$$

Comparing (11) and (10) one can see that the convergence of the randomized ideal multiplication algorithm is much slower in the case of coprime ideals. This implies that one should avoid multiplying the coprime ideals, as well as the ideals I_{M_1} and I_{M_2} with substantially different M_1 and M_2. Namely, one should avoid computing $J_{h+1}=K_hI_{M^{(m-h-1)}}$. Furthermore, the method for construction of $I_{M^{(h)}}$ based on (2) should be replaced with a more efficient approach.

These problems will be addressed in the following subsections, where some modifications to the layered interpolation algorithm will be presented.

B. Constructing a basis for $\widehat{J}_{0,\rho_0}$

In most cases matrix $M^{(m)}$ contains many non-zero rows. We propose to employ a modified Lee-O'Sullivan algorithm to construct a Gröbner basis of $\widehat{J}_{0,\rho_0}$, and use it as a starting point of the layered algorithm. For brevity, the layer index m will be dropped in the subsequent derivations.

Assume that the set G_0 contains all re-encoding locators found in layer L. Then after the re-encoding transformation the basis polynomials (9) can be represented as

$$\widehat{B}_0(x,z) = \prod_{i\in\{L\setminus R\}_x}(x-x_i), \qquad (12)$$

$$\widehat{B}_s(x,z) = u_s(x)\frac{\phi(x)z+f(x)-w_0(x)}{\psi^{(h)}(x)}a_s(x,z) = \left(\sum_{(x_i,y_j)\in R\setminus G_0}\frac{y_j\lambda_{R\setminus G_0}(x)}{\phi'(x_i)(x-x_i)} - \sum_{(x_i,y_j)\in G_0\setminus R}\frac{y_j\lambda_{G_0\setminus R}(x)}{u_0'(x_i)(x-x_i)} + \sum_{(x_i,y_j)\in R\cap G_0} y_j\frac{\frac{\lambda_{R\setminus G_0}(x)}{\phi'(x_i)}-\frac{\lambda_{G_0\setminus R}(x)}{u_0{}'(x_i)}}{x-x_i}\right)u_s(x)a_s(x,z), \quad s\geq 1.$$

where $a_s(x,z)=\prod_{t=1}^{s-1}(\phi(x)z+f(x)-w_t(x))$ and $\lambda_A(x)=\prod_{(x_i,y_j)\in A}(x-x_i)$. Observe that these expressions are simpler than the ones given in [14].

These polynomials should be processed as described in Section II-B using $\prec_{-1}$. The termination criterion at step 4 should be changed to

$$\operatorname{xdeg}\widehat{B}_s(x,z) = sk-\tau_m. \qquad (13)$$

This approach requires $\rho\approx\theta$ calls to the multidimensional Euclidean algorithm, while each coprime ideal multiplication operation in (2) requires usually more than one call to it, and θ ideal multiplications should be performed.

C. Incrementing root multiplicities

The analysis given in Section IV-A implies that computing a Gröbner basis of $J_{h+1}=K_hI_{M^{(m-h-1)}}$ and $K_h=J_h^2$ via ideal multiplication requires approximately the same number of operations. This is not efficient, since most matrices $M^{(m-h-1)}$ are sparse. Therefore we propose to implement this step using IIA [3]. Before applying it, one should process the points $(x_i,y_j)\in R\cap L_h$. Without re-encoding, one could increment the multiplicities of these roots by multiplying $K_{h,\rho_h'}$ (for some suitable ρ_h') by $T_h=\left[\psi^{(h)}(x), y-w_{h,0}(x)\right]$, where $w_{h,0}(x_i)=y_j$ for all $(x_i,y_j)\in R\cap L_h$. Observe that this basis is a Gröbner one with respect to $\prec_{k-1}$. After the change of variables it becomes $\left[\psi^{(h)}(x),\phi(x)z+f(x)-w_{h,0}(x)\right]=\left[\psi^{(h)}(x),\phi(x)z\right]$. Eliminating the common factor $\psi^{(h)}(x)$, one obtains the module $\widehat{T}_h=[1,\prod_{(x_i,y_j)\in R,M_{i,j}^{(h)}=0}(x-x_i)z]$. Hence, $\widehat{J}_{h+1}'=\widehat{K}_{h,\rho_h'}\widehat{T}_h$, where $\widehat{K}_{h,\rho_h'}$ is the module obtained from $K_{h,\rho_h'}$ by re-encoding. Since the basis of $\widehat{T}_h$ contains just two very simple polynomials, the randomized multiplication algorithm given by (4) can handle this case very efficiently.

```
HYBRIDINTERPOLATION($M$)
 1  $R \leftarrow \{\arg\max_{(x_i,y_j)} M_{i,j}\}, |R| = k$
 2  $m \leftarrow \lfloor \log\max_{i,j} M_{i,j} \rfloor$
 3  Let  $M = \sum_{h=0}^{m} 2^h M^{(h)}, M_{i,j}^{(h)} \in \{0,1\}$
 4  $\mathcal{B} \leftarrow$ INITIALMODULE($M^{(m)}, R$)
 5  for $h \leftarrow m-1$ to $0$
 6  do $\mathcal{B} \leftarrow$ MERGE($\mathcal{B}, \mathcal{B}$)
 7      $\mathcal{G} \leftarrow (1, \prod_{(x_i,y_i)\in R, M_{i,j}^{(h)}=0}(x-x_i)z)$
 8      $\mathcal{B} \leftarrow$ MERGE($\mathcal{B}, \mathcal{G}$)
 9      for $(x_i,y_j) \notin R$
10      do for $j_1, j_2 : j_1+j_2+1 = \sum_{t=h}^{m} 2^{t-h} M_{i,j}^{(t)}$
11          do $\rho \leftarrow |\mathcal{B}|-1$
12              $\sigma_t \leftarrow$ HASSE($B_t, x_i, y_j, j_1, j_2, R$), $t = 0,..,\rho$
13              $t_0 \leftarrow \arg\min_{t:\sigma_t\neq 0} \operatorname{wdeg}_{(1,-1)} \operatorname{LT} Q_t(x,z)$
14              if $t_0 = \rho$
15                then $B_{\rho+1} \leftarrow B_\rho\phi(x)z, \rho \leftarrow \rho+1$
16                     $\sigma_\rho \leftarrow$ HASSE($B_\rho, x_i, y_j, j_1, j_2, R$)
17              $B_t(x,z) \leftarrow B_t(x,z) - \frac{\sigma_t}{\sigma_{t_0}} B_{t_0}(x,z), t \neq t_0$
18              $B_{t_0}(x,z) \leftarrow B_{t_0}(x,z)(x-x_i)$
19  return $\mathcal{B}$
```

Fig. 1. Construction of Gröbner basis of I_M

The remaining points $(x_i,y_j)\notin R: M_{i,j}^{(m-h-1)}=1$ can be processed with IIA. That is, for all $j_1,j_2: j_1+j_2+1=2W_{i,j}^{(h)}+M_{i,j}^{(m-h-1)}$, one can apply the transformations given by steps 1–4 in Section II-C to the Gröbner basis of $\widehat{J}_{h+1}'$. Observe that one needs to construct (before re-encoding) polynomials $Q_t(x,y), t=0,\ldots,\rho_{h+1}$, which constitute a Gröbner basis both of module $I_{W^{(h+1)},\rho_{h+1}}$ and zero-dimensional ideal $I_{W^{(h+1)}}$. This implies that $\operatorname{xdeg}Q_{\rho_{h+1}}(x,y)=0$. After re-encoding this constraint transforms to (13). However, it may happen that IIA selects at step 2 the last polynomial as the smallest one, and multiplies it by $x-\alpha$ at step 4. Obviously, this will break the constraint on xdeg. To avoid this problem one should append to the basis the last polynomial multiplied by $\phi(x)z$ (y without re-encoding). Observe also that employing the re-encoding transformation requires one to use modified interpolation points given by (8).

D. Proposed algorithm

Figure 1 summarizes the proposed algorithm. Subroutine $InitialModule$ was described in Section IV-B. For details of $Merge$, see [13]. Function $Hasse(B_t,x_i,y_j,j_1,j_2,R)$ computes the appropriate Hasse derivative of polynomial B_t if $i\notin\{R\}_x$, and modified Hasse derivative (see [3, Eq. (52)]) otherwise.

V. Complexity analysis

The complexity of the proposed algorithm is dominated by the last layer. According to [13], module squaring requires $O(nr^3(a\log(r\sqrt{n/k})\log(nr)+b(n-\sqrt{nk})))$ operations, where r is the maximum root multiplicity, a and b are some constants. The computational cost of lines 9–18 of algorithm

TABLE I
INTERPOLATION TIME FOR $(255, 239)$ CODE, S

	$\max M_{i,j} = 3$		$\max M_{i,j} = 6$		$\max M_{i,j} = 9$		$\max M_{i,j} = 12$		$\max M_{i,j} = 15$	
E_b/N_0, dB	HA	IIA	HA	IIA	HA	IIA	HA	IIA	HA	IIA
6.0	0.0077	0.0139	0.0161	0.0273	0.0394	0.0690	0.1835	0.1684	0.2160	0.3884
6.2	0.0063	0.0139	0.0122	0.0264	0.0356	0.0651	0.1447	0.1572	0.1934	0.3576
6.4	0.0060	0.0141	0.0109	0.0264	0.0316	0.0628	0.0921	0.1422	0.1648	0.3150
6.6	0.0059	0.0138	0.0108	0.0259	0.0281	0.0588	0.0603	0.1274	0.1519	0.2798
6.8	0.0057	0.0139	0.0099	0.0243	0.0252	0.0555	0.0476	0.1188	0.1382	0.2575
7.0	0.0052	0.0136	0.0089	0.0235	0.0225	0.0514	0.0403	0.1086	0.1302	0.2376

TABLE II
INTERPOLATION TIME FOR $(63, 31)$ CODE, S

	$\max M_{i,j} = 3$		$\max M_{i,j} = 6$		$\max M_{i,j} = 9$		$\max M_{i,j} = 12$		$\max M_{i,j} = 15$	
E_b/N_0, dB	HA	IIA	HA	IIA	HA	IIA	HA	IIA	HA	IIA
4.8	0.0014	0.0050	0.0194	0.0419	0.0695	0.2009	0.2847	0.6376	0.6845	1.5568
5.0	0.0016	0.0056	0.0183	0.0450	0.0732	0.2166	0.2986	0.6565	0.6915	1.5969
5.2	0.0017	0.0059	0.0165	0.0471	0.0770	0.2141	0.3207	0.6538	0.7070	1.6057
5.4	0.0018	0.0064	0.0145	0.0473	0.0830	0.2167	0.3199	0.6640	0.7108	1.5572
5.6	0.0021	0.0066	0.0135	0.0492	0.0860	0.2237	0.3037	0.6386	0.7121	1.4701

in Figure 1 can be estimated as $O(n^2r^4)$. Hence the overall complexity of the hybrid interpolation algorithm is given by $O(n^2r^4)$. This is less than the complexity of IIA, which is $O(n^2r^5)$.

VI. NUMERIC RESULTS

The proposed algorithm was implemented in C++ programming language, and its efficiency was assessed by computer simulations. Transmission of the binary image of Reed-Solomon codes over the AWGN channel with BPSK modulation was considered. The entries of the root multiplicitiy matrix were computed as $M_{i,j} = \lfloor \lambda P_{i,j} \rfloor$, where P is the corresponding probability matrix.

Table I illustrates the complexity of the interpolation step for the case of $(255, 239)$ code over $GF(2^8)$ based on IIA and the proposed one (HA). Re-encoding was used in both cases. It can be seen that the hybrid interpolation algorithm provides more than two times complexity reduction, and the gain increases with SNR. The reason is that for well-selected λ the density of $M^{(h)}, h < m$, decreases with SNR, reducing thus the amount of computation at steps 7–18. However, as it can be seen from Table II, for the low-rate code this effect is outweighted by the need to process high degree polynomials with the multidimensional Euclidean algorithm, which is invoked by $Merge$.

VII. CONCLUSION

In this paper a novel hybrid bivariate interpolation algorithm for algebraic soft decision decoding of Reed-Solomon codes was proposed. This algorithm combines the layered interpolation algorithm [2] with the re-encoding transformation, iterative interpolation and Lee-O'Sullivan algorithms. Numeric results indicate that it provides more than two times complexity reduction compared to the case of iterative interpolation algorithm.

ACKNOWLEDGEMENTS

This work was partially supported by the grant MK-1976.2011.9 of the President of Russia and by the Saint-Petersburg government research grant for students.

REFERENCES

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

[2] V. Miloslavskaya and P. Trifonov, "Fast interpolation in algebraic soft decision decoding of Reed-Solomon codes," in *Proc. of IEEE R8 Int. Conf. on Computational Technologies in Electrical and Electronics Engineering*, 2010, pp. 65–69.

[3] R. Koetter, J. Ma, and A. Vardy, "The re-encoding transformation in algebraic list-decoding of Reed-Solomon codes," *IEEE Trans. On Inf. Theory*, vol. 57, no. 2, February 2011.

[4] K. Lee and M. E. O'Sullivan, "An interpolation algorithm using Gröbner bases for soft-decision decoding of Reed-Solomon codes," in *Proc. of IEEE Int. Symposium on Inf. Theory*, 2006, pp. 2032 – 2036.

[5] R. Koetter, "Fast generalized minimum-distance decoding of algebraic-geometry and Reed-Solomon codes," *IEEE Trans. On Inf. Theory*, vol. 42, no. 3, May 1996.

[6] R. R. Nielsen and T. Hoholdt, "Decoding Reed-Solomon codes beyond half the minimum distance," in *Proc. of the Int. Conf. on Coding Theory and Cryptography*. Mexico: Springer-Verlag, 1998, pp. 221–236.

[7] J. Jiang and K. R. Narayanan, "Algebraic soft-decision decoding of Reed-Solomon codes using bit-level soft information," *IEEE Trans. On Inf. Theory*, vol. 54, no. 9, September 2008.

[8] F. Parvaresh and A. Vardy, "Multiplicity assignments for algebraic soft-decoding of Reed-Solomon codes," in *Proc. of IEEE Int. Symposium on Inf. Theory*, June 2003, p. 205.

[9] H. Das and A. Vardy, "Multiplicity assignments for algebraic soft-decoding of Reed-Solomon codes using the method of types," in *Proc. of IEEE Int. Symposium on Inf. Theory*, 2009.

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

[11] T. Sauer, "Polynomial interpolation of minimal degree and Gröbner bases," in *Proc. of the Int. Conf. "33 Years of Gröbner Bases"*, 1998.

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

[13] P. Trifonov, "Efficient interpolation in the Guruswami-Sudan algorithm," *IEEE Trans. on Inf. Theory*, vol. 56, no. 9, pp. 4341–4349, September 2010.

[14] J. Ma and A. Vardy, "A complexity reducing transformation for the Lee—O'Sullivan interpolation algorithm," in *Proc. of IEEE Int. Symposium on Inf. Theory*, 2007, pp. 1986 – 1990.