On the Average Complexity of Reed-Solomon Algebraic List Decoders

Yuval Cassuto and Jehoshua Bruck
California Institute of Technology
Department of Electrical Engineering
E-mail: {ycassuto,bruck}@paradise.caltech.edu

***Abstract*— We study the error dependent interpolation cost of Reed-Solomon (RS) list-decoders. Words corrupted by low weight errors require lower degree interpolation polynomials. By giving different upper bounds we quantitatively link the interpolation cost to the weight of the instantaneous channel error. To take advantage of this phenomenon, we propose modifications to interpolation algorithms that translate reduced interpolation cost to reduced running time. This new property immediately implies improved average case decoding time, which can be useful in various applications. We validate the tightness of the bounds by comparing them to actual interpolation costs observed in a RS list-decoder. We also consider soft-decision decoders and experimentally reveal their rather different behavior.**

***Index Terms*— List decoding, Reed-Solomon, Average case complexity, Interpolation cost**

I. Introduction

The advent of algebraic list-decoding algorithms for Reed-Solomon codes by Guruswami and Sudan [4] has created an immense interest in their theoretical understanding and practical implementation. Decoding widely used codes such as Reed-Solomon codes beyond their error correction bound had since become an option not to be overlooked by the coding theory community. Despite its unquestionable virtue, the Guruswami-Sudan (GS) method enfolds two main negative artifacts that needed further study. These are the *non-unique decoding* and the high (polynomial) *time complexity* of decoding. Insights about both issues appeared in a multitude of works, and this paper will offer more understanding about the latter. More specifically, we will hereafter study the *average case* complexity of decoding, where the average is taken over the number of channel errors. The average case analysis hinges upon the observation that the number of interpolation coefficients, defined as *interpolation cost*, and consequently the decoding complexity, depends on the instantaneous number of channel errors and not only on the designed worst case number of errors handled by the decoder. That gap between design worst case and average case complexity is of utmost significance, especially for channels in which the case of "many errors" is relatively rare. Analyzing the average complexity of a decoder is of prime practical importance. When designing decoders, taking into account only the worst case decoding complexity is suboptimal. In [2], which discusses a traditional Reed-Solomon decoder, a similar observation (experimental in that case) of error-weight dependent decoding time is made. That observation then motivates a buffered implementation of the decoder that dramatically increases its effective speed. More recently, [3] seeks to improve the average case complexity of the RS algebraic soft decision decoder [5], by using a layered decoder whose decode time depends on the instantaneous channel noise. The main purposes of the current paper are first to understand and quantify that error dependent behavior and second to utilize it for improved average case decoding time. The analytical portion of our results apply only to the hard-decision decoder whereas the algorithmic and experimental results consider both soft and hard decision decoding. We characterize interpolation polynomials for low-weight error words and use that characterization to upper bound the number of interpolation coefficients as a function of the error weight. For the extreme case of no errors the number of interpolation coefficients is shown to be a constant fraction of the worst case number of coefficients, with the constant being the rate of the code. For nonzero numbers of errors, closed form bounds are given for the number of coefficients, and a method to obtain tighter bounds (not in closed form) is then provided. We evaluate the bounds on a sample RS code and show that they are tight in the sense that the bound values are attained, and that happens not just in the worst case, but in almost all instances. In the algorithmic domain, we show how to guarantee good average case interpolation complexity, a property that conventional interpolation algorithms fail to encompass. Finally, we empirically study the interpolation costs in an algebraic soft-decision RS decoder and see its different behavior compared to the hard-decision case.

[1]This work was supported in part by the Lee Center for Advanced Networking at the California Institute of Technology.

II. Definitions, Notations and GS Decoding

A codeword C from an $[n,k]$ Reed-Solomon (RS) code is the evaluation of a degree $k-1$ or less message polynomial $f(x)$ on n distinct points of GF(q), $\{\alpha_1,\ldots,\alpha_n\}$. Let E be an error vector of Hamming weight e over the same alphabet GF(q). The received word R is defined as $R=C+E$ over GF(q) arithmetic. In the Guruswami-Sudan algorithm, the received word is used to interpolate a bivariate polynomial $Q(x,y)$. To achieve a large correction radius, $Q(x,y)$ should be the minimal weighted degree bivariate polynomial that satisfies the following $n\binom{m+1}{2}$ constraints: $D_{r,s}Q(\alpha_i,R_i)=0$ for $i=\{1,\ldots,n\}$ and $\{(r,s):r+s<m\}$. m is a decoder parameter called the interpolation multiplicity. $D_{r,s}Q(\alpha,\beta)$ is the Hasse derivative of x-order r and y-order s, evaluated on the point $x=\alpha,y=\beta$ (see III-A below and [7].) By formulating the interpolation as a system of homogeneous linear equations it has been observed that $n\binom{m+1}{2}+1$ coefficients are sufficient to make $Q(x,y)$ satisfy the above constraints. We denote by C_w this worst case number of interpolation coefficients, so $C_w=n\binom{m+1}{2}+1$. C_w will be later called the worst case *interpolation cost* of the RS (n,k,m) decoder. The key yield from that decoding scheme is that a sufficient condition to correct t errors is $m(n-t)>d_{1,k-1}(C_w)$, where $d_{1,k-1}(J)$ is the minimal $(1,k-1)$ weighted degree of a bivariate polynomial with J coefficients. Throughout the paper, we will assume that the monomials of the interpolating polynomials are ordered by nondecreasing $(1,k-1)$ weighted degrees, with revlex tie-breaking, i.e $x^{(k-1)s}$ precedes $x^{(k-1)(s-1)}y$.

III. Interpolation Polynomials for Low-Weight Errors

In this section we develop tools for bounding interpolation cost given an error weight. To this end we have introduced the worst case interpolation cost C_w which is determined by the decoder parameters n,k,m. For the sake of the forthcoming analysis we will define the error-weight dependent interpolation cost $C_\epsilon(e)$ as the number of interpolation coefficients required given an error weight e. Note that $C_\epsilon(e)$ is not a function of e; different interpolation costs are possible for a given weight e. It is also important to emphasize that this section is non-constructive in nature with existential arguments that are aimed toward proving bounds.

A. Hasse derivatives

In this subsection we define the Hasse derivatives of a bivariate polynomial and present some basic facts about them that will become useful later in the section.

Definition 1—(r,s) Hasse derivative: The (r,s) Hasse derivative of a polynomial $Q(x,y)$ denoted $D_{r,s}Q(x,y)$ is defined as

$$D_{r,s}Q(x,y)=\sum_{i,j}\binom{i}{r}\binom{j}{s}a_{i,j}x^{i-r}y^{j-s}$$

where $a_{i,j}$ is the coefficient of x^iy^j in $Q(x,y)$.
Hasse derivatives owe their use in RS decoding to the following fact

$$D_{r,s}Q(\alpha,\beta)=\mathrm{coeff}_{x^ry^s}Q(x+\alpha,y+\beta)$$

We now turn to state (without proof) the product rule for Hasse derivatives.

Lemma 2— (The Hasse Derivative product rule):
the Hasse derivative of a product of L polynomials

$$D_{r,s}\left[\prod_{i=1}^{L}Q_i\right]=\sum_{\substack{r_1+\cdots+r_L=r\\ s_1+\cdots+s_L=s}}\prod_{i=1}^{L}D_{r_i,s_i}Q_i$$

From lemma 2 we get the following lemma.

Lemma 3: If $Q(\alpha,\beta)=0$, then for every $\{r,s:r+s<m\}$, $D_{r,s}Q(\alpha,\beta)^m=0$, where $D_{r,s}Q(\alpha,\beta)^m$ is the r,s Hasse derivative of $Q(x,y)^m$ evaluated at (α,β).
Proof: lemma 2 states

$$D_{r,s}Q(x,y)^m=\sum_{\substack{r_1+\cdots+r_m=r\\ s_1+\cdots+s_m=s}}\prod_{i=1}^{m}D_{r_i,s_i}Q(x,y)$$

since $r+s=\sum_{i=1}^{m}(r_i+s_i)<m$, for every assignment to $r_1,s_1,\ldots,r_m,s_m$ at least one of the pairs (r_i,s_i) equals $(0,0)$. That means every product in the sum contains at least one factor $D_{0,0}Q(x,y)=Q(x,y)$. Substituting $(x,y)=(\alpha,\beta)$ the right hand side evaluates to zero.

B. Closed form upper bound on interpolation cost

Theorem 4: Let E be an error vector of weight e and let $\{j_1,j_2,\ldots,j_e\}$ be the error locations. Then there exists an interpolation polynomial whose last monomial has (x,y)-degree of (em,m). This polynomial can be explicitly written as

$$Q(x,y)=[(y-f(x))(x-\alpha_{j_1})(x-\alpha_{j_2})\cdots(x-\alpha_{j_e})]^m \tag{1}$$

Proof:

$$D_{r,s}Q(x,y)=D_{r,s}\left[(y-f(x))^m(x-\alpha_{j_1})^m\cdots(x-\alpha_{j_e})^m\right]=$$

$$=\sum_{\substack{r_0+r_1+\cdots+r_e=r\\ s_0+s_1+\cdots+s_e=s}}D_{r_0,s_0}(y-f(x))^m\prod_{i=1}^{e}D_{r_i,s_i}(x-\alpha_{j_i})^m=$$

$$=\sum_{r_0+r_1+\cdots+r_e=r}D_{r_0,s}(y-f(x))^m\prod_{i=1}^{e}D_{r_i,0}(x-\alpha_{j_i})^m$$

If (r,s) satisfy $r+s<m$, then obviously $r_0+s<m$ and $r_i<m$ for $i=\{1,\ldots,e\}$. Therefore by lemma 3, any product in the sum will have both a factor of $y-f(x)$ and factors of $x-\alpha_{j_i}$, for all $i=\{1,\ldots,e\}$. This establishes that $D_{r,s}Q(\alpha_i,R_i)=0$ for both the correct symbols and the corrupted symbols. □

Taking the polynomial structure of (1) with some straight forward monomial counting we get the following corollary.

Corollary 5: Let $\Delta=em+(k-1)m$ and $r=\Delta \bmod (k-1)$. For errors of weight e we have the following bound

$$C_\epsilon(e)\le\frac{\Delta^2}{2(k-1)}+\frac{\Delta}{2}+\frac{r(k-r-1)}{2(k-1)}+m+1 \quad (2)$$

C. The no errors case

Theorem 6: Let $\rho=\frac{k}{n}$ be the rate of the code. When the received word R is a codeword, the interpolation cost $C_\epsilon(0)$ satisfies

$$C_\epsilon(0)\le\lceil\rho C_w\rceil$$

Proof:

Define $v=k-1$. When R is a codeword, R can be interpolated to the bivariate polynomial $Q(x,y)=(y-f(x))^m$. The monomial with the largest $(1,v)$-degree in $Q(x,y)$ is y^m and this degree equals mv.

Lemma 7: The number of coefficients C in a $(1,v)$-degree ordered bivariate polynomial of $(1,v)$-degree mv or less is $v\binom{m+1}{2}+m+1$.

Proof:

$$C=|(i,j):i+vj\le mv|=v\binom{m+1}{2}+m+1$$

Now using lemma 7 we get

$$C=v\binom{m+1}{2}+m+1=k\binom{m+1}{2}-\frac{1}{2}(m+1)(m-2)$$

Substituting $k\binom{m+1}{2}=\rho C_w-\rho$:

$$C=\rho C_w-\rho-\frac{1}{2}(m+1)(m-2)$$

$\frac{1}{2}(m+1)(m-2)\ge-1$ and so

$$C\le\rho C_w-\rho+1\le\lceil\rho C_w\rceil$$

Since $C_\epsilon(0)\le C$ the theorem follows. □

D. Tighter bounds for higher weight errors

When e is large, bounds on the interpolation cost can still be obtained, though using (2) may not be the best choice. For such cases we can use the following theorem.

Theorem 8: Let E be an error vector of weight e and let $\{j_1,j_2,\ldots,j_e\}$ be the error locations. Then there exists an interpolation polynomial whose last monomial has (x,y)-degree of $(d_x,m'+d_y)$ where $m'\le m$ is an integer and (d_x,d_y) is the lowest degree of a polynomial $P(x,y)$ that satisfies $e\binom{m+1}{2}+(n-e)\binom{m-m'+1}{2}$ interpolation constraints. This polynomial will be of the form

$$Q(x,y)=(y-f(x))^{m'}P(x,y)$$

Proof: We find a (minimal $(1,k-1)$-degree) polynomial $P(x,y)$ that satisfies the following constraints. For the corrupted locations $\{j_1,j_2,\ldots,j_e\}$, we require the usual interpolation constraints $D_{r,s}P(\alpha_{j_i},R_{j_i})=0$ for every $(r,s):r+s<m$. For the $n-e$ uncorrupted locations we require fewer such constraints- $D_{r,s}P(\alpha_j,R_j)=0$ for every $(r,s):r+s<m-m'$. Since for the corrupted symbols $P(x,y)$ alone satisfies all interpolation requirements, $Q(x,y)$ obviously does so too. As for the uncorrupted symbols we write

$$D_{r,s}Q(x,y)=\sum_{\substack{r_1+r_2=r\\ s_1+s_2=s}}D_{r_1,s_1}(y-f(x))^{m'}D_{r_2,s_2}P(x,y)$$

Splitting the sum to two disjoint intervals

$$D_{r,s}Q(x,y)=\sum_{\substack{r_1+r_2=r\\ s_1+s_2=s\ :\\ r_1+s_1<m'}}D_{r_1,s_1}(y-f(x))^{m'}D_{r_2,s_2}P(x,y)+$$

$$+\sum_{\substack{r_1+r_2=r\\ s_1+s_2=s\ :\\ r_1+s_1\ge m'}}D_{r_1,s_1}(y-f(x))^{m'}D_{r_2,s_2}P(x,y)=$$

$$=\sum_{\substack{r_1+r_2=r\\ s_1+s_2=s\ :\\ r_1+s_1<m'}}(\text{left addend})+\sum_{\substack{r_1+r_2=r\\ s_1+s_2=s\ :\\ r_2+s_2\le r+s-m'}}(\text{right addend})$$

The left sum is zero by lemma 3 and the right sum is zero since $r+s-m'<m-m'$ and the $(r_2,s_2):r_2+s_2<m-m'$ Hasse derivatives of $P(x,y)$ vanish on the uncorrupted locations by construction. □

Notes:

(1) theorem 4 is a special case of theorem 8 with $m'=m$ and $P(x,y)$ univariate in x. In general, for each e, m' can be freely chosen to find the best bound on the interpolation cost $C_\epsilon(e)$ achieved by $Q(x,y)$.

(2) In the interesting cases $1\le m'<m$, for each uncorrupted location the composition polynomial $(y-f(x))^{m'}P(x,y)$ satisfies more interpolation constraints relative to the sum of constraints satisfied by the individual components $(y-f(x))^{m'}$ and $P(x,y)$. $(y-f(x))^{m'}$ satisfies $\binom{m'+1}{2}$ constraints, $P(x,y)$ satisfies $\binom{m-m'+1}{2}$ and as proved in theorem 8, $Q(x,y)$ satisfies $\binom{m+1}{2}$. These numbers reflect a difference of $m'(m-m')$.

IV. Interpolation costs for a sample RS code

In section III bounds are given for the error-weight dependent interpolation costs. Here we wish to explore the tightness of these bounds by interpolating received words induced by different channel errors and comparing the observed interpolation costs to the bounds above. For that task we implemented a GS decoder and ran it on a $[31, 15]$ RS code. We took $m = 3$ which allows correcting 9 errors and has a worst case interpolation cost of $C_w = 187$. The results are summarized in table I below. Each row reflects a value of e and the columns compare observed results to the bounds. The columns tagged *observed* are the maximum, average and minimum interpolation costs used by the decoder. These numbers were generated using repeating runs ($\sim 10^5$ per e) with random errors. For $e \leq 6$, no interpolation costs smaller than the closed form bound of corollary 5 were observed. For $e = 7$ the bound is attained in almost all instances, with few exceptions of up to a difference of 2. That is the case also for $e = 8$, only that theorem 8 is used to find an improved bound over corollary 5. For $e = 9$ the best upper bound for the interpolation cost is C_w.

				observed		
#errors e	worst case C_w	closed form (2)	improved [Theorem 8]	max	average	min
0	187	88	-	88	88	88
1	187	100	-	100	100	100
2	187	112	-	112	112	112
3	187	124	-	124	124	124
4	187	136	-	136	136	136
5	187	149	-	149	149	149
6	187	164	-	164	164	164
7	187	179	-	179	178.95	177
8	187	194	183 , $m' = 1$	183	182.97	179
9	187	209	187 , $m' = 0$	187	186.93	184

TABLE I

Interpolation costs for the $(31, 15)$ RS code with $m = 3$

V. From interpolation cost to decoding complexity

In the preceding sections we argued that in many cases the interpolation cost is significantly lower than the worst case C_w. That immediately means factorization algorithms would run faster in low cost instances. However, the most computationally expensive part of the decoder is the interpolation algorithm. Unfortunately, a reduced interpolation cost does not automatically provide reduced running time of interpolation algorithms. Admittedly, we will see that accepted interpolation algorithms do not translate the savings in coefficients to savings in running time. That is true even in light of the fact that these algorithms do eventually output the lowest degree interpolation polynomials. This situation is unfortunate since the decoder fails to benefit from the worst-case/average-case gap that was pointed out earlier. We will examine such behaviors of two interpolation algorithms in the case of reduced interpolation cost. We will subsequently suggest variations on them to improve their average-case running time.

A. Gaussian elimination interpolation

By formulating the interpolation problem as a system of homogeneous linear equations, Gaussian elimination stands out as a natural straight forward algorithm. This interpolation method is not the most efficient that exists and we present it only to illustrate the connection between interpolation cost and running time. A naive way to use Gaussian elimination is to start with a $(C_w - 1) \times C_w$ matrix and perform full Gaussian elimination. The number of rows being the number of interpolation constraints and the number of columns is the worst case interpolation cost. Since that matrix is underdetermined, at termination we are guaranteed to reveal linearly dependent columns which result in coefficients of an interpolating polynomial. To analyze the running time of the above procedure, we will approximate the dimensions of the matrix by $C_w \times C_w$. It is well known that the running time of Gaussian elimination on a $c \times c$ matrix approaches $\frac{2}{3}c^3$ finite field operations (plus lower order terms $o(c^3)$) [9, ch. IV]. This follows from

$$2\sum_{k=1}^{c}\sum_{j=k+1}^{c}(c-k+1) \rightarrow \frac{2}{3}c^3$$

Thus using straight forward Gaussian elimination will consume $\frac{2}{3}C_w^3$ finite field operations, regardless of the actual interpolation cost of the decoding instance. By using a simple variation on that process we can save considerably in the total number of field operations. When a shorter interpolation polynomial exists, some of the columns in the matrix will not participate in the interpolation. Exploiting that, a row operation should be performed on a column index only if the columns to its left are linearly independent. This replaces the row operation on the full row vector performed in Gaussian elimination. An even more obvious modification is stopping the process at the first time linearly dependent columns are revealed. If we denote $c = C_w$, $c' = C_\epsilon(e)$ and $\gamma = \frac{c'}{c}$, then the running time of the modified Gaussian elimination will be

$$2\sum_{k=1}^{c'}\sum_{j=k+1}^{c}(c'-k+1) \rightarrow \frac{2}{3}cc'^2 + \frac{1}{3}c'^2(c-c') =$$

$$= \frac{2}{3}C_w^3\left(\frac{3}{2}\gamma^2 - \frac{1}{2}\gamma^3\right)$$

and that yields a $\frac{3}{2}\gamma^2 - \frac{1}{2}\gamma^3$ factor of saving.

B. Standard interpolation algorithm

Now we wish, for the same purpose of average case analysis, to consider the standard, most efficient interpolation algorithm used in RS list decoding. This algorithm and its variants are intensively studied in the literature [8], [6], [1] and more. A detailed discussion of this algorithm is beyond the scope of this paper so we will present a simplified view of it that will suffice to convey our ides. A (rough) sketch of the algorithm is found in figure 1. *min* and *argmin* are taken with respect to the standard monomial ordering on the highest degree monomials of each j. The algorithm iterates on $C_w - 1$ constraints and

Initialize
$Q_j := y^j,\ \forall j \in \{0, \ldots, L\}$ `// L is a bound on the decoder list size`
for $i := 1$ **to** $C_w - 1$ `// interpolation constraints`
(1) $\delta_j^{(i)} :=$ discrepancy of Q_j with respect to constraint i
$j^* := \text{argmin}(j : \delta_j^{(i)} \neq 0)$
forall j with nonzero $\delta_j^{(i)}$
(2) for $j \neq j^*$ update Q_j with no change in degree
for j^* update Q_{j^*} with degree increment
output Q_j with minimal degree

Fig. 1. Standard interpolation algorithm (sketch)

in each iteration performs operations *(1)* and *(2)* on (at most) $L + 1$ polynomials, each with no more than C_w coefficients. Therefore the worst case running time is LC_w^2 times the running time of *(1)* and *(2)*. We next argue that the running time will not be significantly better in cases when the final interpolation cost is small. The reason being the computation load is dominated by operations on non minimal polynomials. Even if a final Q_{j^*} polynomial ends up having low cost, the algorithm does not know the identity of that j^* in advance and has to perform computations on all Q_j which have higher costs. It also does not apriori know the final cost required and thus cannot exclude polynomials with higher costs. Consequently, this fast interpolation algorithm will have an average case running time not better than that of the worst case. To fix that undesirable behavior, we modify the algorithm in a way that discrepancy calculations and polynomial updates are performed only on polynomials whose coefficient counts are guaranteed to be at most the final interpolation cost. This can be done by reordering the same operations above, with no increase in worst case running time. The price we have to pay is increased space complexity used to store previous versions of polynomials Q_j. We can think of the modified algorithm as a relayed version of the standard algorithm where each time the leading candidate (the min degree polynomial) is sequentially updated, until a better candidate is found. On its way, before it is updated with degree increase, the best candidate stores its coefficients and discrepancies to allow for future candidates to "catch up" with their updates. In figure 2 the modified algorithm is given. *mem-lookup* in *(*)* refers to the action of looking for a stored polynomial Q_j that had nonzero discrepancy on i_{j^*}. Instead of a full correctness proof of the algorithm, we will state the central facts that show it is equivalent to the standard algorithm, only without wasteful operations on polynomials whose degrees exceed that of the final returned polynomial.

(1) Discrepancy calculations and polynomial updates are performed only on polynomials with degrees lower or equal than the final interpolation polynomial.

(2) The first Q_j whose pointer i_j reaches an index i is the lowest degree polynomial that satisfies constraints $1, \ldots, i-1$. Therefore, the stored polynomials will always be the lowest degree polynomials that satisfy $1, \ldots, i-1$ but not i.

(3) If mem-lookup fails for Q_j on constraint i_j, it is equivalent to Q_j being the lowest degree polynomial with nonzero discrepancy on i_j.

(4) The polynomial whose pointer i_j first reaches C_w satisfies all constraints and is the minimal to achieve that.

For every constraint, at most one polynomial is stored and each of these has at most $C_\epsilon(e)$ coefficients. Thus in this non-optimized formulation, the amount of memory required for coefficient storage is bounded by γC_w^2.

Initialize
$Q_j := y^j,\ \forall j \in \{0, \ldots, L\}$
$i_j := 0,\ \forall j \in \{0, \ldots, L\}$ `// constraint pointer for each` j
$j^* := 0$ `//` $j^* = \text{argmin}_j Q_j$
while $i_{j^*} < C_w - 1$ `// while no` Q_j `satisfies all constraints`
$i_{j^*}++$
find $\delta_{j^*}^{(i_{j^*})}$
if $\delta_{j^*}^{(i_{j^*})} = 0$ **continue**
(*) **mem-lookup** $(\Delta[j, i_{j^*}],\ \mathcal{Q}[j, i_{j^*}])$ `// look for stored poly`
if (found) **update** Q_{j^*} with no change in degree
else
store $(\Delta[j^*, i_{j^*}],\ \mathcal{Q}[j^*, i_{j^*}]) \leftarrow (\delta_{j^*}^{(i_{j^*})},\ Q_{j^*})$
update Q_{j^*} with degree increment
$j^* := \text{argmin}_j Q_j$ `// proceed with the best candidate`

output Q_{j^*}

Fig. 2. Interpolation algorithm with improved average running time

VI. Interpolation cost in soft decision decoders

The bounds presented thus far apply to GS decoders which have a fixed interpolation multiplicity m. They do not apply to the weighted interpolation used by Koetter and Vardy's soft decision decoder that was shown to correct more errors when

soft inputs are available. In this section we will examine another aspect of soft decision decoders, their error dependent interpolation costs. Since soft decision decoders surrender their fixed multiplicity property, none of the bounds above apply to them. Moreover, when the decoder inputs are soft symbols, different ways exist to define the instantaneous channel error upon which the interpolation cost may depend. The difficulty of analytic treatment of the soft decision case arises from the fact that the interpolation cost depends on the interpolation multiplicities which in turn depend on the channel error in a non-simple fashion. The bounds obtained for the hard decision case used the structure of the interpolation polynomial endowed by the fixed multiplicity m. It is therefore conjectured that the soft decision decoder will not enjoy as favorable interpolation cost behavior, and consequently will have higher average case decoding complexity, even if it is designed for the same worst case cost as the hard decision decoder. To support that conjecture we veer to the experimental realm.

A. *Simulation results for soft-decision decoder*

For the $[31, 15]$ RS code of section IV, we simulated soft decoding over a channel whose description follows. We regard the 32 alphabet symbols as integers lying on a ring of circumference 32. The noise is taken to be an additive (modulo 32) iid random process, denoted $N = \{N_1, \ldots, N_n\}$. For simplicity we take the pdf of N_i to have a bounded support $(-1, 1)$. This property implies that at most two symbols will be assigned nonzero interpolation multiplicities by the Koetter-Vardy algorithm. The decoder we used has a worst case interpolation cost identical to that of the hard decision decoder we used in section IV, $C_w = 187$. It is thus interesting to compare the interpolation cost exhibited by the soft-decision decoder to that of the hard decision decoder. To have a ground for comparison, we will plot the interpolation cost as a function of the number of "hard" errors e caused by the channel. This number can be recovered by $e = |\{N_i : |N_i| > 0.5\}|$. In figure 3 the HD and SD average interpolation costs are plotted as a function of the number of errors. Each point on the graphs was obtained from an order of 10^5 runs. We see that for low error weights the SD decoder requires higher interpolation costs compared to the HD decoder. For high weights it is more efficient but only slightly. Another difference is that for a given e, the SD decoder exhibited a high variability in the interpolation cost, whereas in table I the HD decoder is shown to have very predictable costs. Both the relative flatness in figure 3 and the cost variability indicate that in SD decoding, the dependence of the interpolation costs on the error weight is rather weak, contrary to the HD case. Once we have good average case decoders, this trade-off between correcting more errors and increasing the average case complexity should be taken into account when designing a decoder.

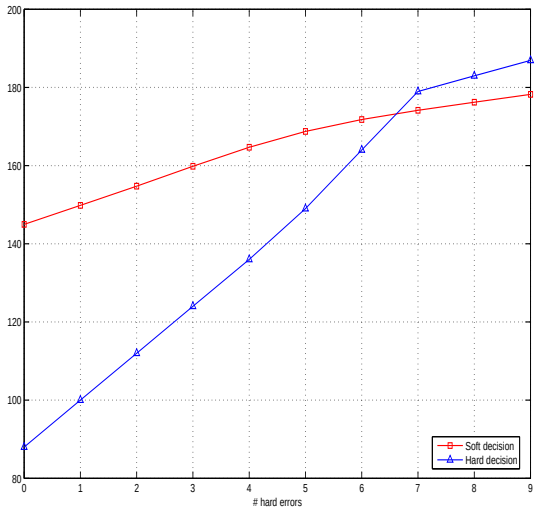

Fig. 3. SD-HD comparison, Average interpolation cost

REFERENCES

[1] A. Ahmed, R. Koetter, and N. Shanbhag. VLSI architectures for soft-decision decoding of reed-solomon codes. *Submitted to IEEE-Trans-VLSI*.

[2] E. Berlekamp and R. McEliece. Average-case optimized buffered decoders. In *The impact of processing techniques on communications*, pages 145–158, Martinus Nijhoff publishers, The Netherlands, 1985. NATO Advanced science institutes.

[3] W. Gross, F. Kschischang, R. Koetter, and P. Gulak. Applications of algebraic soft-decision decoding of reed-solomon codes. *Submitted to IEEE-Trans-Comm, July 2003*.

[4] V. Guruswami and M. Sudan. Improved decoding of reed-solomon and algebraic-geometric codes. *IEEE-Trans-IT*, 45:1755–1764, 1999.

[5] R. Koetter and A. Vardy. Algbraic soft-decision decoding of reed-solomon codes. *IEEE-Trans-IT*, 49(11):2809–2825, November 2003.

[6] R. Koetter and A. Vardy. A complexity reducing transformation in algebraic list decoding of reed-solomon codes. In *proc. of ITW*, Paris, 2003.

[7] R.J McEliece. The guruswami-sudan algorithm for decoding reed-solomon codes. Technical Report IPN progress report 42-153, JPL, http://www.ipnpr.jpl.nasa.gov/progress_report/42-153/, 2003.

[8] R.R Nielsen and T. Hoholdt. Decoding reed-solomon codes beyond half the minimum distance. *Cryptography and Related Areas, J. Buchmann et al. eds. Springer-Verlag 2000*, pages 221–236, 2000.

[9] L. Trefethen and D. Bau. *Numerical linear algebra*. Society for industrial and applied mathematics, Philadelphia PA, 1997.