A Low-Complexity Method for Chase-Type Decoding of Reed-Solomon Codes

Jason Bellorado
Link-A-Media Devices
Santa Clara, CA, 95054, USA
Email: jbellorado@link-a-media.com

Aleksandar Kavčić
Division of Engineering and Applied Sciences
Harvard University
Cambridge, MA, 02138, USA
Email: kavcic@deas.harvard.edu

***Abstract*— In this work we present a low-complexity implementation of Chase-type decoding of Reed-Solomon Codes. In such, we first use the soft-information available at the channel output to construct a test-set of 2^η vectors, equivalent in all except the $\eta << n$ least reliable coordinate positions. We then give an interpolation procedure to construct a set of 2^η bivariate polynomials, with the roots of each specified by its corresponding test-vector. Here, test-vector similarity is exploited to share much of the required computation. Finally, we obtain the candidate message from the single z-linear factor of each bivariate polynomial. Although we provide an expression for the direct computation of each candidate message, the complexity of repeating this computation for each interpolation polynomial is prohibitive. We, thus, also present a reduced-complexity factorization (RCF) method to select a single polynomial that, with high probability, contains the correctly decoded message in its z-linear factor. Although suboptimal, the loss in performance of RCF decreases rapidly with increasing code length. We provide extensive simulation results showing that a significant performance increase over traditional hard-decision decoding is achievable with a comparable computational complexity (as implemented with the *Berlekamp-Massey Algorithm*).**

I. Introduction

The first practical decoding procedure for Reed-Solomon (RS) codes was developed by Berlekamp in 1968 [1]. The hard-decision decoding (HDD) algorithm is capable of successfully decoding any vector with no more than $d_{\min}/2$ errors, where $d_{\min}$ is the minimum distance of the code. Even decades after its development, variants of Berlekamp's original formulation (*Berlekamp-Massey (B-M) Algorithm* [2], *Euclid Algorithm* [3]) are, overwhelmingly, the most implemented RS decoding procedures in modern communications.

Despite a myriad of ensuing work, it was not until 1997 that Sudan [4] demonstrated the achievability of RS decoding beyond $d_{\min}/2$. These performance gains, however, are achievable with a complexity exponential in the decoding radius expansion. Sudan's work was generalized by Köetter and Vardy [5] in 2000, in which reliability information was used to achieve larger coding gains with a comparable complexity.

Although these procedures are widely perceived as the first algorithms capable of decoding RS codes beyond $d_{\min}/2$, in 1972 Chase [6] had developed a decoding method, non-specific to the type of code, that allows for an increase in the decoding radius of any existing HDD procedure. In traditional HDD, the *maximum a-posteriori* (MAP) hard-decision vector is used as input to the decoding algorithm. In Chase decoding, however, reliability information is used to develop a set of test-vectors, each of which is applied to the decoding algorithm. Decoding beyond $d_{\min}/2$ is, therefore, possible using traditional decoding techniques provided any test-vector falls within the decoding radius of the utilized algorithm. This performance increase, however, is gained at the expense of a linear increase in complexity with the test-set cardinality.

In this work we consider Chase-type decoding of RS codes using sets of test-vectors that are equivalent in all but a small number of indices. Using a point-by-point interpolation algorithm, we detail a procedure that exploits this similarity to significantly reduce the complexity of the first decoding step (interpolation). A factorization procedure is then developed that constructs a candidate message from each interpolation polynomial. Due to its high complexity, we also detail a reduced-complexity method that selects the single polynomial most likely decode correctly. Although suboptimal, we provide simulation results that show a decreasing loss in performance with increasing code length.

The remainder of this paper is organized as follows: In Section II we discuss our utilized notation and the construction of the test-set. In Section III we detail the decoding procedure, consisting of test-set modification, interpolation, and polynomial factorization. A reduced-complexity factorization procedure is also provided here. We present a complexity comparison to existing RS decoding methods in Section IV and simulation results in Section V. A brief conclusion is provided in Section VI.

II. Background and Notation

A. Coding and Modulation

We consider RS codes of length n, dimension k, and, due to their maximum distance separable nature [7], minimum distance $d_{\min} = n - k + 1$. The codes considered are over the Galois Field of size 2^q, denoted as GF(2^q), and, therefore, all algebraic operations are performed within this field. We specify this code by $\mathcal{C}_q(n,k)$, which maps a k-tuple of message symbols, $\mathbf{m} = (m_0, m_1, \ldots, m_{k-1})$, $m_i \in$ GF(2^q), to an n-tuple codeword $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$, $c_i \in$ GF(2^q), where $\mathbf{c} \in \mathcal{C}_q(n,k)$. Here, we consider the *evaluation-map* encoding of $(m_0, m_1, \ldots, m_{k-1})$, in which a message polynomial $m(z) = m_0 + m_1 z + \cdots + m_{k-1} z^{k-1}$ is evaluated at n distinct elements of GF(2^q). If an ordered set of field elements is given by $\mathcal{F} = \{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\}$, $\alpha_i \in$ GF(2^q),

1-4244-0504-1/06/$20.00 ©2006 IEEE

then the codeword is formed as, $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1}) = (m(\alpha_0), m(\alpha_1), \ldots, m(\alpha_{n-1}))$. We note that our choice of an *evaluation-map* encoding is because it lends itself to an insightful interpretation of the utilized interpolation algorithm. All results presented, however, hold for any encoding method.

The modulation considered is Binary Phase Shift Keying (BPSK) and, thus, all symbols transmitted are in the set $\mathcal{S} = \{-d, +d\}$, where d controls the transmitted power (and, thus, the signal-to-noise ratio). Since binary modulation is used, we map each coordinate of the codeword to the length-q transmission using a bijection $\mathcal{M} : \text{GF}(2^q) \rightarrow \mathcal{S}^q$. The transmitted vector is, thus, formed as, $\mathbf{x} = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{n-1})$, $\mathbf{x}_i = (x_{iq}, x_{iq+1}, \ldots, x_{(i+1)q-1}) = \mathcal{M}(c_i)$. Since the bijection utilized will not affect our results, it is not further discussed.

B. Obtaining the Test-Set

In this study, we consider an Additive White Gaussian Noise (AWGN) channel, which, for a single codeword, is represented by the equation $\mathbf{r} = \mathbf{x} + \mathbf{n}$. Here, $\mathbf{n}$ is a length-(nq) vector composed of independent, identically distributed, zero-mean, unit-variance, Gaussian random variables. The MAP estimate of the transmitted vector $\mathbf{x}$ is given by,

$$\hat{\mathbf{x}}^{(\text{MAP})} = \arg \max_{\mathbf{x} \in \mathcal{S}^{nq}} p(\mathbf{x}|\mathbf{r}). \tag{1}$$

Because we consider an AWGN channel,

$$p(r_i|\mathbf{x}) = p(r_i|x_i), \tag{2}$$

and the MAP estimate $\hat{\mathbf{x}}^{(\text{MAP})} = (\hat{x}_0^{(\text{MAP})}, \hat{x}_1^{(\text{MAP})}, \ldots, \hat{x}_{nq-1}^{(\text{MAP})})$ is obtained on a symbol-by-symbol basis as,

$$\hat{x}_i^{(\text{MAP})} = \arg \max_{x_i \in \mathcal{S}} p(x_i|r_i). \tag{3}$$

From the inverse bijection, $\mathcal{M}^{-1} : \mathcal{S}^q \rightarrow \text{GF}(2^q)$, we obtain the hard-decision vector $\mathbf{y}^{\text{HD}} = (y_0^{\text{HD}}, y_1^{\text{HD}}, \ldots, y_{n-1}^{\text{HD}})$ from the MAP symbol estimates as, $y_i^{\text{HD}} = \mathcal{M}^{-1}(\hat{x}_{iq}^{(\text{MAP})}, \hat{x}_{iq+1}^{(\text{MAP})}, \ldots, \hat{x}_{(i+1)q-1}^{(\text{MAP})})$.

In addition to the hard-decision vector $\mathbf{y}^{\text{HD}}$, we also require a secondary hard-decision vector $\mathbf{y}^{\text{2HD}}$. From the above discussion, it is clear that we may write,

$$\hat{\mathbf{x}}^{(\text{MAP})} = (\hat{\mathbf{x}}_0^{(\text{MAP})}, \hat{\mathbf{x}}_1^{(\text{MAP})}, \ldots, \hat{\mathbf{x}}_{n-1}^{(\text{MAP})}) \tag{4}$$

where,

$$\hat{\mathbf{x}}_i^{(\text{MAP})} = \arg \max_{\mathbf{x}_i \in \mathcal{S}^q} p(\mathbf{x}_i|\mathbf{r}_i). \tag{5}$$

Similarly, we define the secondary MAP estimate vector as,

$$\hat{\mathbf{x}}^{(\text{MAP2})} = (\hat{\mathbf{x}}_0^{(\text{MAP2})}, \hat{\mathbf{x}}_1^{(\text{MAP2})}, \ldots, \hat{\mathbf{x}}_{n-1}^{(\text{MAP2})}) \tag{6}$$

where each component is the second-best decision,

$$\hat{\mathbf{x}}_i^{(\text{MAP2})} = \arg \max_{\mathbf{x}_i \in \mathcal{S}^q, \mathbf{x}_i \neq \hat{\mathbf{x}}_i^{(\text{MAP})}} p(\mathbf{x}_i|\mathbf{r}_i). \tag{7}$$

We obtain the secondary hard-decision vector $\mathbf{y}^{\text{2HD}} = (y_0^{\text{2HD}}, \ldots, y_{n-1}^{\text{2HD}})$ as $y_i^{\text{2HD}} = \mathcal{M}^{-1}(\hat{\mathbf{x}}_i^{(\text{MAP2})})$.

Using $\mathbf{y}^{\text{HD}}$, $\mathbf{y}^{\text{2HD}}$, and $\mathbf{r}$, we compute a figure-of-merit for each coordinate position as,

$$\gamma_i = \frac{p(c_i = y_i^{\text{2HD}}|r_i)}{p(c_i = y_i^{\text{HD}}|r_i)} \leq 1. \tag{8}$$

This value gives a measure of the reliability of each coordinate position ($\gamma_i \approx 1$ indicates a high probability that the hard-decision is in error and the secondary hard-decision is correct). Given a positive integer η, we define the set of the lowest reliability indices (those that maximize γ_i) as $\mathcal{I} = \{i_1, i_2, \ldots, i_\eta\}$, and the remaining coordinate positions as $\bar{\mathcal{I}} = \{\bar{i}_1, \bar{i}_2, \ldots, \bar{i}_{n-\eta}\} = \{0, 1, \ldots, n-1\} \setminus \mathcal{I}$. The test-set consists of all vectors $\mathbf{y}_i = (y_{i,0}, \ldots, y_{i,n-1})$, with

$$y_{i,j} \in \begin{cases} \{y_j^{\text{HD}}\} & if \quad j \in \bar{\mathcal{I}} \\ \{y_j^{\text{HD}}, y_j^{\text{2HD}}\} & if \quad j \in \mathcal{I}. \end{cases}$$

The test-set, denoted as $\mathbf{Y}$, has cardinality $2^{|\mathcal{I}|} = 2^\eta$.

III. Low-Complexity Chase (LCC) Decoding

In traditional Chase decoding, each $\mathbf{y}_i$ is applied to an HDD algorithm to produce a set of candidate codewords $\hat{\mathcal{C}}$. The decoded codeword is then chosen as the element of $\hat{\mathcal{C}}$ of highest *a-posteriori probability*. Clearly, this methodology requires running the HDD algorithm 2^η times for each transmission, which is extremely complex for large values of η. Since the test-vectors are equivalent in all but (at most) η indices, we exploit this similarity and share much of the decoding computation. For this, we first modify each test-vector.

A. Test-Set Modification

To reduce the decoding complexity, the test-set is first modified by *zeroing-out* k entries of each vector. For this, we express any test-vector $\mathbf{y} \in \mathbf{Y}$ as the sum of a codeword vector $\mathbf{c}$ and an error vector $\mathbf{e}$ as,

$$\mathbf{y} = \mathbf{c} + \mathbf{e}. \tag{9}$$

For any set $\mathcal{J} = \{j_1, j_2, \ldots, j_k\} \subseteq \{0, 1, \ldots, n-1\}$, we determine a codeword $\Psi = (\psi_0, \ldots, \psi_{n-1}) \in \mathcal{C}_q(n, k)$ with the property,

$$\psi_i = y_i \quad for\ every \quad i \in \mathcal{J}. \tag{10}$$

Determining Ψ is equivalent to an erasures-only decoding [7] of $\mathbf{y}$ with coordinate positions $\bar{\mathcal{J}} = \{0, 1, \ldots, n-1\} \setminus \mathcal{J}$ erased and, thus, is always possible. Adding Ψ to $\mathbf{y}$ yields,

$$\begin{aligned} \mathbf{y}' &= \mathbf{y} + \Psi \\ &= \mathbf{c}' + \mathbf{e} \end{aligned} \tag{11}$$

where $\mathbf{c}' = \mathbf{c} + \Psi \in \mathcal{C}_q(n, k)$ due to the linearity of RS codes. Because $\mathbf{y}$ and $\mathbf{y}'$ have the same error vector, using either in the decoding process is equivalent. We, however, exploit the fact that $\mathbf{y}'$ is zero in k coordinate positions to reduce the decoding complexity.

We choose $\mathcal{J}$ as any set of indices satisfying

$$\mathcal{J} \subseteq \bar{\mathcal{I}}. \tag{12}$$

Because the test-vectors are equivalent on the indices in $\bar{\mathcal{I}}$, and because Ψ depends only on the value of the coordinate positions in $\mathcal{J}$, this choice of $\mathcal{J}$ forces Ψ to be the same for all vectors in $\mathbf{Y}$. We, thus, modify all test-vectors as $\mathbf{y}_i' = \mathbf{y}_i + \Psi$, which maintains their equivalence on the indices in $\bar{\mathcal{I}}$, and denote the resulting set as $\mathbf{Y}'$. Although any set $\mathcal{J}$ satisfying (12) will suffice, we later discuss a more judicious choice.

ALGORITHM 1: COMMON-ELEMENT INTERPOLATION

INPUT
$(\alpha_i, y_i^{\text{HD}}, \psi_i),\ \forall i \in \bar{\mathcal{J}} \cap \bar{\mathcal{I}} = \mathcal{B}_c$
INITIALIZATION
$v_i = v(\alpha_i),\ i \in \mathcal{B}_c$
$\mathcal{G} = \{g_0(x,z), g_1(x,z)\} = \{1, z\}$
INTERPOLATION
FOR $i \in \mathcal{B}_c$
compute:
$(\alpha_i, \tilde{y}_i) = (\alpha_i, (y_i^{\text{HD}} + \psi_i)/v_i)$
$g_0(\alpha_i, \tilde{y}_i)$
$g_1(\alpha_i, \tilde{y}_i)$
choose:
$f(x,z) = \arg\min_{g(x,z)\in\mathcal{G}}\{\deg_{(1,\text{-}1)}(g(x,z)),\ g(\alpha_i, \tilde{y}_i) \neq 0\}$
$g(x,z) = \mathcal{G} \setminus \{f(x,z)\}$
update:
$g(x,z) = g(x,z) + (g(\alpha_i, \tilde{y}_i)/f(\alpha_i, \tilde{y}_i)) \cdot f(x,z)$
$f(x,z) = (x + \alpha_i) \cdot f(x,z)$
$\mathcal{G} = \{g_0(x,z), g_1(x,z)\} = \{g(x,y), f(x,y)\}$
END FOR
OUTPUT
$\mathcal{G}_c = \{(g_0(x,z), g_1(x,z))\}$

ALGORITHM 2: UNCOMMON-ELEMENT INTERPOLATION

INPUT
$(\alpha_i, y_i^{\text{HD}}, y_i^{\text{2HD}}, \psi_i),\ \forall i \in \mathcal{I},\ \mathcal{G}_c$
INITIALIZATION
$\mathcal{G}_0 = \{\mathcal{W}_0^{(0)}\} = \mathcal{G}_c$
$v_{i_j} = v(\alpha_{i_j}),\ j \in \{1, 2, \ldots, \eta\}$
INTERPOLATION
FOR $j = 1, 2, \ldots, \eta$
compute:
$(\tilde{y}_{i_j}^{\text{HD}}, \tilde{y}_{i_j}^{\text{2HD}}) = ((y_{i_j}^{\text{HD}} + \psi_{i_j})/v_{i_j}, (y_{i_j}^{\text{2HD}} + \psi_{i_j})/v_{i_j})$
FOR $m = 0, 1, \ldots, 2^{j-1} - 1$
$\mathcal{G} = \mathcal{W}_m^{(j-1)}$
compute:
$g_0(\alpha_i, \tilde{y}_i^{\text{HD}}),\ g_1(\alpha_i, \tilde{y}_i^{\text{HD}})$
$g_0(\alpha_i, \tilde{y}_i^{\text{2HD}}),\ g_1(\alpha_i, \tilde{y}_i^{\text{2HD}})$
choose:
$f_{\text{HD}}(x,z) = \arg\min_{g(x,z)\in\mathcal{G}'}\{\deg_{(1,\text{-}1)}(g(x,z)),\ g(\alpha_i, \tilde{y}_i^{\text{HD}}) \neq 0\}$
$f_{\text{2HD}}(x,z) = \arg\min_{g(x,z)\in\mathcal{G}'}\{\deg_{(1,\text{-}1)}(g(x,z)),\ g(\alpha_i, \tilde{y}_i^{\text{2HD}}) \neq 0\}$
$g_{\text{HD}}(x,z) = \mathcal{G}' \setminus \{f_{\text{HD}}(x,z)\}$
$g_{\text{2HD}}(x,z) = \mathcal{G}' \setminus \{f_{\text{2HD}}(x,z)\}$
update:
$g_{\text{HD}}(x,z) = g_{\text{HD}}(x,z) + \big(g_{\text{HD}}(\alpha_i, \tilde{y}_i^{\text{HD}})/f_{\text{HD}}(\alpha_i, \tilde{y}_i^{\text{HD}})\big) \cdot f_{\text{HD}}(x,z)$
$g_{\text{2HD}}(x,z) = g_{\text{2HD}}(x,z) + \big(g_{\text{2HD}}(\alpha_i, \tilde{y}_i^{\text{2HD}})/f_{\text{2HD}}(\alpha_i, \tilde{y}_i^{\text{2HD}})\big) \cdot f_{\text{2HD}}(x,z)$
$f_{\text{HD}}(x,z) = (x + \alpha_i) \cdot f_{\text{HD}}(x,z)$
$f_{\text{2HD}}(x,z) = (x + \alpha_i) \cdot f_{\text{2HD}}(x,z)$
$\mathcal{W}_{2m}^{(j)} = (g_{\text{HD}}(x,z), f_{\text{HD}}(x,z))$
$\mathcal{W}_{2m+1}^{(j)} = (g_{\text{2HD}}(x,z), f_{\text{2HD}}(x,z))$
END FOR
END FOR
OUTPUT
$\mathcal{G}_\eta = \{\mathcal{W}_0^{(\eta)}, \mathcal{W}_1^{(\eta)}, \ldots, \mathcal{W}_{2^\eta-1}^{(\eta)}\}$

B. Polynomial Interpolation

The first step in decoding the elements of $\mathbf{Y}'$ is to interpolate each vector into a bivariate polynomial, the utility of which is given by the following *Lemma*,

Lemma 1: Consider a hard-decision vector $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1})$ and a bivariate polynomial $Q(x,z)$ with z-degree at most one, for which the following holds,

$$Q(x,z) = 0 \quad for\ (x,z) \in \{(\alpha_0, y_0), \ldots, (\alpha_{n\text{-}1}, y_{n\text{-}1})\} \quad (13)$$

If $m(x) = m_0 + \cdots + m_{k-1}x^{k-1}$ is a message polynomial whose *evaluation-map* codeword $\mathbf{c} = (m(\alpha_0), \ldots, m(\alpha_{n-1}))$ has Hamming distance no greater than $d_{\min}/2$ from $\mathbf{y}$, then $(z - m(x))$ divides $Q(x,z)$. □

Proof: See [8]. ■

Our objective is to obtain a polynomial $Q_i(x,z)$ corresponding to test-vector $\mathbf{y}'_i = (y'_{i,0}, y'_{i,1}, \ldots, y'_{i,n-1}) \in \mathbf{Y}'$ for which,

$$Q_i(\alpha_j, y'_{i,j}) = 0,\ j \in \{0, 1, \ldots, n-1\}. \quad (14)$$

Before presenting the procedure we use to obtain these polynomials, we first make the following observation. Since the z-degree of each $Q_i(x,z)$ is at most one, we may express it as $Q_i(x,z) = q_{i,0}(x) + z \cdot q_{i,1}(x)$. Assume that we have a second polynomial $\tilde{Q}_i(x,z) = \tilde{q}_{i,0}(x) + z \cdot \tilde{q}_{i,1}(x)$ for which the constraints of (14) hold only for $j \in \bar{\mathcal{J}}$. By defining $v(x)$,

$$v(x) = \prod_{i=1}^{k}(x - \alpha_{j_i}), \quad (15)$$

the polynomial $Q_i(x,z)$, obeying the constraints of (14) for all coordinate positions, may be constructed as

$$Q_i(x,z) = \tilde{q}_{i,0}(x) \cdot v(x) + z \cdot \tilde{q}_{i,1}(x). \quad (16)$$

Thus, our interpolation algorithm constructs $\tilde{Q}_i(x,z)$ and forms $Q_i(x,z)$ using (16). This is the complexity reduction achieved by the test-set modification.

In light of this observation, we need only interpolate the $(n-k)$ coordinate positions in $\bar{\mathcal{J}}$, for which we use a point-by-point interpolation algorithm [9]. Denoting the set of the first m interpolated indices as $\mathcal{P}^{(m)} = \{l_1, \ldots, l_m\}$, the algorithm, during the m^{th} step, constructs an intermediate polynomial $Q_i^{(m)}(x,z)$ with roots specified by the elements of $\mathcal{P}^{(m)}$, i.e.

$$Q_i^{(m)}(\alpha_j, y'_{i,j}) = 0 \quad for\ every\ j \in \mathcal{P}^{(m)}. \quad (17)$$

Clearly, the order in which we consider the indices is arbitrary since, at the completion of the algorithm, the polynomial constructed will obey (14) for all coordinates in $\bar{\mathcal{J}}$.

We partition the set $\bar{\mathcal{J}}$ into two sets, $\mathcal{B}_c = \bar{\mathcal{J}} \cap \bar{\mathcal{I}}$ ($|\mathcal{B}_c| = (n-k-\eta)$) and $\mathcal{B}_u = \bar{\mathcal{J}} \cap \mathcal{I} = \mathcal{I}$ ($|\mathcal{B}_u| = \eta$). The irrelevance of the index ordering allows us to first interpolate the positions in $\mathcal{B}_c$, since all test-vectors in $\mathbf{Y}'$ are equivalent in these locations. We provide the *Common Element Interpolation* procedure as **Algorithm 1**[1], the output of which is the pair of polynomials $\mathcal{G}_c$ which obey the constraints of (14) only for the indices in $\mathcal{B}_c$.

The test-vectors are not equivalent in the remaining η positions in $\mathcal{B}_u$ and, thus, interpolation is slightly more difficult. We know, however, that for all $j \in \mathcal{B}_u$, the z-value is one of the following two forms

- $y'_{i,j} = \psi_j + y_j^{\text{HD}} = \tilde{y}_j^{\text{HD}}$
- $y'_{i,j} = \psi_j + y_j^{\text{2HD}} = \tilde{y}_j^{\text{2HD}}$.

[1]The notation $\deg_{(1,\text{-}1)}(g(x,z))$ is used to denote the (1,-1)-weighted degree of $g(x,z)$ (see [9]).

ALGORITHM 3: SELECTION OF POLYNOMIAL $Q^*(x,z)$

INPUT

$\tilde{\mathcal{Q}} = \{\tilde{Q}_1(x,z), \tilde{Q}_2(x,z), \ldots, \tilde{Q}_{2^\eta}(x,z)\}$

COMPUTATION

Compute in parallel for $i = 1, 2, \ldots, 2^\eta$

$p_i = |\{\alpha_j;\ j \in \bar{\mathcal{J}},\ \tilde{q}_{i,1}(\alpha_j) = 0\}|$

$d_{0,i} = \deg(\tilde{q}_{i,0}(x)) - p_i$

$d_{1,i} = \deg(\tilde{q}_{i,1}(x)) - p_i$

$w_i = \sum_{j=1}^{\eta} \log(p(r_j|y_{i,j}))$

SELECTION

1) $d_0^* = \min_{i\in\{0,1,\ldots,2^\eta-1\}} d_{0,i}$

2) $\Gamma = \{i;\ d_{i,0} < d_{i,1},\ d_{0,i} = d_0^*\}$

3) $i^* = \arg\max_{i\in\Gamma} w_i$

4) $Q^*(x,z) = Q_{i^*}(x,z)$

OUTPUT

$Q^*(x,z)$

We, therefore, perform interpolation using a binary tree structure (**Algorithm 2**), the root of which is the set $\mathcal{G}_0 = \mathcal{G}_c$. The interpolation of each, successive, index in $\mathcal{B}_u$ is represented by a progression of one level deeper into the tree, in which we construct the set $\mathcal{G}_j$ from $\mathcal{G}_{j-1}$ by interpolating index i_j. Here, the polynomial pairs $\mathcal{W}_{2m}^{(j)}$ and $\mathcal{W}_{2m+1}^{(j)}$ are obtained by interpolating the pair $\mathcal{W}_m^{(j-1)}$ through the data points $(\alpha_{i_j}, \tilde{y}_{i_j}^{\text{HD}})$ and $(\alpha_{i_j}, \tilde{y}_{i_j}^{\text{2HD}})$, respectively. The size of the polynomial set, thus, doubles after considering each index. The end result of interpolation is the set $\mathcal{G}_\eta$, with each of its 2^η polynomials corresponding to a distinct test-vector in $\mathbf{Y}'$. Either element of $\mathcal{W}_i^{(\eta)}$ is chosen as $\tilde{Q}_i(x,z)$, which comprise the set $\tilde{\mathcal{Q}}$.

Although **Algorithm 2** appears to be complex, we note that much of the computation required to interpolate $\mathcal{W}_{2m}^{(j)}$ and $\mathcal{W}_{2m+1}^{(j)}$ from $\mathcal{W}_m^{(j-1)}$ may be shared. The computational savings result from the common starting polynomial pair and the equivalence of the x-value of the interpolation points (see [8] for a complete description).

C. Full Polynomial Factorization

Using the set $\tilde{\mathcal{Q}}$ produced by **Algorithm 2**, we construct $\mathcal{Q} = \{Q_1(x,z), \ldots, Q_{2^\eta}(x,z)\}$ by applying (16) to each $\tilde{Q}_i(x,z)$ to obtain $Q_i(x,z)$. Since $Q_i(x,z)$ obeys the constraints of (13) for vector $\mathbf{y}'_i$, *Lemma 1* dictates that the associated candidate message polynomial is contained in the single z-linear factor of $Q_i(x,z)$. From the *Fundamental Theorem of Algebra*, because $(z - m_i(x))$ is a factor of $Q_i(x,z)$, $Q_i(x, m_i(x)) = 0$. Using (16), we obtain,

$$m_i(x) = \frac{v(x)\tilde{q}_{i,0}(x)}{\tilde{q}_{i,1}(x)}. \tag{18}$$

We, thus, apply (18) to each $Q_i(x,z)$ to obtain its associated candidate message polynomial $m_i(x)$, and select the decoded message $m^*(x)$ as that of highest *a-posteriori probability*. With this methodology, decoding performance equivalent to traditional Chase decoding is achieved (see Figs. 1 and 2).

The computation required to conduct this procedure is significant for large values of η. This is particularly true for the long (large n), high-rate (correspondingly large k) codes that we are primarily interested in. We, thus, next present a reduced-complexity factorization (RCF) procedure.

TABLE I

MAXIMUM MULTIPLICATIONS REQUIRED FOR DECODING

ALGORITHM	$\mathcal{C}_6(63,55)$	$\mathcal{C}_8(255,239)$
HDD using Berlekamp-Massey	1208	7920
Köetter-Vardy (max. multiplicity = 4)	$3.6 \cdot 10^4$	$2.5 \cdot 10^5$
LCC using RCF ($\eta = 2$)	1037	–
LCC using RCF ($\eta = 3$)	1294	–
LCC using RCF ($\eta = 4$)	1831	6806
LCC using RCF ($\eta = 5$)	–	8399
LCC using RCF ($\eta = 6$)	–	11636

D. Reduced-Complexity Factorization (RCF)

The high complexity of full factorization incurs mostly due to the need to repeat the procedure for each element of $\mathcal{Q}$. Here, we present a method to select a single bivariate polynomial $Q^*(x,z) \in \mathcal{Q}$, whose z-linear factor $(z - m^*(x))$ contains the decoded message polynomial $m^*(x)$. Extensive simulations are provided to attest to the accuracy of RCF.

We first make a more judicious choice of $\mathcal{J}$, which was previously chosen as any set satisfying (12). However, we now select $\mathcal{J}$ as the k indices that minimize γ_i (see (8)). Since selecting $\mathcal{I}$ requires finding the largest η values of γ_i, we extend this to finding the largest $(n-k)$ values to produce $\bar{\mathcal{J}}$. We next provide intuition leading to RCF.

Although candidate message polynomial $m_i(x)$ is expressed as a quotient in (18), it must be a valid message polynomial and, thus, have degree no larger than $(k-1)$. The denominator $\tilde{q}_{i,1}(x)$ must, therefore, be completely canceled by the numerator $v(x)\tilde{q}_{i,0}(x)$. Here, we make the assumption that $\tilde{q}_{i,1}(x)$ has no irreducible factors of degree greater than one (we have observed this to be true). We, thus, express $\tilde{q}_{i,1}(x)$ as

$$\tilde{q}_{i,1}(x) = \zeta \cdot \prod_{\alpha\in\mathcal{A}_i} (x+\alpha) \cdot \prod_{\beta\in\mathcal{B}_i} (x+\beta), \tag{19}$$

where $\mathcal{A}_i$ and $\mathcal{B}_i$ are the sets of all roots of $\tilde{q}_{i,1}(x)$ shared with $v(x)$ and $\tilde{q}_{i,0}(x)$, respectively. Since the set of roots of $v(x)$ is $\mathcal{R} = \{\alpha_{j_1}, \ldots, \alpha_{j_k}\}$, any root of $\tilde{q}_{i,1}(x)$ that is not in $\mathcal{R}$ must be in $\mathcal{B}_i$. We, thus, estimate $|\mathcal{B}_i|$ as the number of roots of $\tilde{q}_{i,1}(x)$ not falling in $\mathcal{R}$ (since their are only $(n-k)$ possibilities), which we denote by p_i. We, then, estimate $|\mathcal{A}_i|$ by $(\deg(\tilde{q}_{i,1}(x)) - p_i)$.

Our interest in $|\mathcal{A}_i|$ is based on our choice of the indices in $\mathcal{J}$. From *Lemma 1*, the *evaluation-map* encoding of $m_i(x)$ must have Hamming distance no greater than $d_{\min}/2$ from $\mathbf{y}'_i$, which is zero for all indices in $\mathcal{J}$. Roots of $v(x)$ canceled by $\tilde{q}_{i,1}(x)$ correspond to hard-decision errors in $\mathcal{J}$, which are the most reliable indices. We, therefore, use our estimate of $|\mathcal{A}_i|$ as a metric to determine the most probable $Q_i(x,z)$ of producing a correctly decoded codeword (**Algorithm 3**). With $Q^*(x,z)$ selected according to **Algorithm 3**, we apply (18) to obtain the decoded message polynomial.

IV. COMPLEXITY COMPARISON TO EXISTING TECHNIQUES

Although a complete complexity analysis is not given (see [8]), our intention is to give results, in terms of the maximum

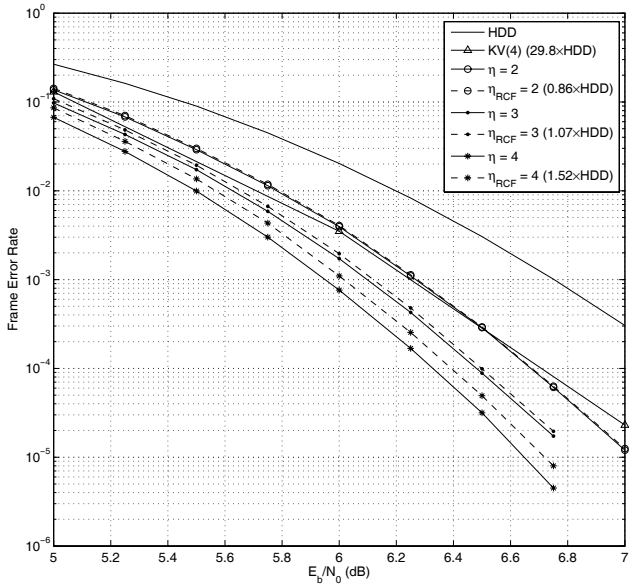

Fig. 1. FER of $\mathcal{C}_6(63, 55)$ for HDD, K-V (max. multiplicity = 4), and LCC decoding ($\eta = 2, 3, 4$) for an AWGN channel. The decoding complexity is indicated as a function of the complexity of HDD (using B-M) (Table I).

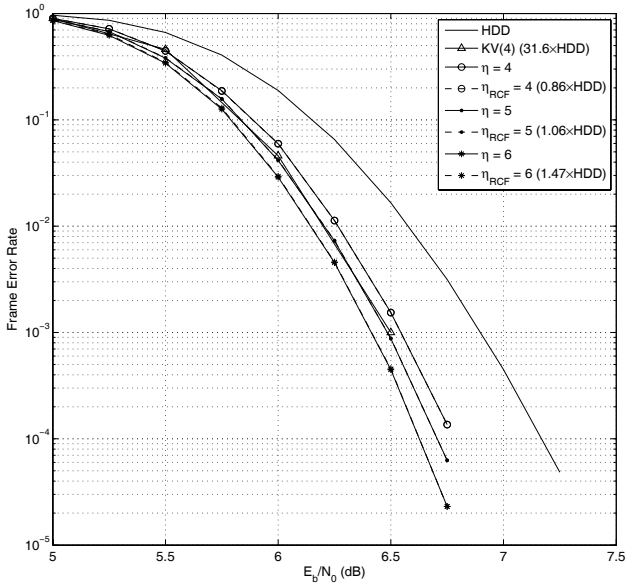

Fig. 2. FER of $\mathcal{C}_8(255, 239)$ for HDD, K-V (max. multiplicity = 4), and LCC decoding ($\eta = 4, 5, 6$) for an AWGN channel. The decoding complexity is indicated as a function of the complexity of HDD (using B-M) (Table I).

number of multiplications required to decode, to make a comparison to existing decoding techniques. Complexity bounds for *Test-Set Modification*, *Common* and *Uncommon Element Interpolation*, and RCF are given as follows;

- $Mults_{\text{Test-Set Modification}} \leq 3(n-k)^2$
- $Mults_{\text{Common}} \leq 3(n-k-\eta)^2 + (n-k-\eta)(\eta+3)$
- $Mults_{\text{Uncommon}} \leq (2^{\eta}-1)(5(n-k)+3) + \eta(n-k+6)$
- $Mults_{\text{RCF}} \leq (n-k) \cdot (2^{(\eta-1)}(n-k)+k)$.

In Table I, we compare the complexity of LCC decoding (using RCF) to HDD (using B-M) and K-V decoding. The figures presented for K-V decoding are assuming a maximum multiplicity of 4, which gives a performance similar to $\eta = 2$ and $\eta = 5$ Chase decoding for $\mathcal{C}_6(63, 55)$ and $\mathcal{C}_8(255, 239)$, respectively. The given complexities are based on the *Reduced-Complexity Interpolation/Factorization* discussions in [9] and, thus, are the lowest values that we are aware of.

Table I shows that the complexity of K-V decoding far exceeds that of HDD or LCC decoding, even for the largest values of η presented. We also note that, for the smallest values of η given, LCC decoding is actually less complex than HDD. We remark that, by selecting $\eta \leq \log_2(n-k)+1$, the LCC complexity is bounded by (about) twice the HDD complexity.

V. Simulation Results

The frame error-rates (FERs) (and associated complexities as a function of HDD complexity) of HDD, K-V, and LCC decoding over an AWGN channel are given in Fig. 1 and Fig. 2 for $\mathcal{C}_6(63, 55)$ and $\mathcal{C}_8(255, 239)$, respectively. We provide curves for three practical values of η (see Section IV) for both full-factorization and RCF. We note that the loss in performance of RCF is increasing with η, a fact that is easily observable in Fig. 1. This behavior, however, is expected since the selection of $Q^*(x, z)$ is more difficult for larger sets $\mathcal{Q}$. The suboptimality of RCF, however, is decreasing with code length. For $\mathcal{C}_8(255, 239)$, there is no observable loss in performance for RCF for η values up to 6. This demonstrates that, for long codes, we may exceed the practical range of values for η without RCF degrading the performance.

VI. Conclusion

We have detailed a low-complexity implementation of RS Chase decoding. In such, interpolation and factorization procedures have been detailed that exploit test-vector similarity to effectively reduce the decoding computation. We have shown that performance similar to (highly complex) K-V decoding is achievable with a decoding complexity comparable to traditional HDD utilizing the B-M Algorithm.

References

[1] E. Berlekamp, "Nonbinary BCH decoding," *IEEE Trans. on Inform. Theory*, vol. IT-14, p. 242, 1968.

[2] J. Massey, "Shift register synthesis and BCH decoding," *IEEE Trans. on Inform. Theory*, vol. IT-15, pp. 122–127, January 1968.

[3] Y. Sugiyama, Y. Kasahara, S. Hirasawa, and T. Namekawa, "A method for solving key equation for goppa codes," *Information and Control*, vol. 27, pp. 87–89, 1975.

[4] M. Sudan, "Decoding of Reed Solomon Codes beyond the error-correction bound," *Journal of Complexity*, pp. 180–193, 1997.

[5] R. Koetter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon Codes," 2000.

[6] D. Chase, "A class of algorithms for decoding block codes with channel measurement information," *IEEE Trans. on Inform. Theory*, vol. IT-18, pp. 170–182, January 1972.

[7] S. Wicker and V. Bhargava, "Reed-Solomon codes and their applications," 1994.

[8] J. Bellorado, *Low-Complexity Soft Decoding Algorithms for Reed-Solomon Codes*. PhD thesis, Harvard University, 2006.

[9] W. J. Gross, F. R. Kschischang, R. Koetter, and P. G. Gulak, "Towards a VLSI architecture for interpolation-based soft-decision Reed-Solomon decoders," *submitted to Journal of VLSI Signal processing*, 2003.