# Low Complexity Soft Decoding Algorithms for Reed-Solomon Codes: Part I - An Algebraic Soft-in-Hard-out Chase Decoder

Jason Bellorado and Aleksandar Kavčić

***Abstract*—In this paper, we present an algebraic methodology for implementing low-complexity, Chase-type, decoding of Reed-Solomon codes of length $n$. In such, a set of $2^\eta$ test-vectors that are equivalent on all except $\eta \ll n$ coordinate positions is first produced. The similarity of the test-vectors is utilized to reduce the complexity of interpolation, the process of constructing a set of polynomials that obey constraints imposed by each test-vector. By first considering the equivalent indices, a polynomial common to all test-vectors is constructed. The required set of polynomials is then produced by interpolating the final $\eta$ dissimilar indices utilizing a binary-tree structure.**

**In the second decoding step (*factorization*) a candidate message is extracted from each interpolation polynomial such that one may be chosen as the decoded message. Although an expression for the direct evaluation of each candidate message is provided, carrying out this computation for each polynomial is extremely complex. Thus, a novel, reduced-complexity, methodology is also given. Although suboptimal, simulation results affirm that the loss in performance incurred by this procedure is decreasing with increasing code length $n$, and negligible for long $(n > 100)$ codes. Significant coding gains are shown to be achievable over traditional hard-in-hard-out decoding procedures (e.g., Berlekamp-Massey) at an equivalent (and, in some cases, lower) computational complexity. Furthermore, these gains are shown to be similar to the recently proposed soft-in-hard-out algebraic techniques (e.g., Sudan, Kötter-Vardy) that bear significantly more complex implementations than the proposed algorithm.**

***Index Terms*—Reed-Solomon codes, Chase decoding, polynomial factorization, polynomial interpolation, reduced-complexity factorization, hard-decision decoding, soft decoding**

## I. Introduction

Since their construction by Irving S. Reed and Gustave Solomon in 1960 [1], Reed-Solomon (RS) Codes have been used ubiquitously in a wide range of applications including, but not limited to, optical and magnetic storage, satellite and deep-space communication, mobile data communication, and spread spectrum systems [2]. Soon after their discovery, the work of Peterson [3] produced an efficient decoding algorithm for non-binary BCH and Reed-Solomon codes that was capable of successfully decoding any received hard-decision vector with Hamming distance no greater than half the code's minimum distance code ($d_{\text{min}}$) from the transmitted codeword. Subsequent works published by Berlekamp [4] and Massey [5] showed that this decoding problem was equivalent to finding the shortest linear feedback shift-register capable of generating a given sequence. They, furthermore, demonstrated the existence of a fast, shift-register based, decoding algorithm that was equivalent to Peterson's algorithm. This procedure has come to be known as the *Berlekamp-Massey (B-M) Algorithm*.

Although a multitude of decoding algorithms have been proposed in the over 45 years since the introduction of RS codes, it was not until the work of Sudan in 1997 [6] that *Bounded-Distance* decoding of RS Codes beyond $d_{\text{min}}/2$ (*the error correction bound*) was shown to be possible. Despite a myriad of ensuring work intended to reduce the complexity of Sudan-type decoding (as well as the generalization of Sudan's algorithm to a soft-in-hard-out decoder proposed by Kötter and Vardy [7] in 2000), it remains orders of magnitude more complex than traditional hard-decision decoding (HDD) techniques (such as Berlekamp-Massey), even when modest gains are sought. Thus, despite the existence of more powerful methodologies, variants of Peterson's original formulation remain, overwhelmingly, the most widely implemented RS decoding procedures in hardware today.

Although the Sudan and Kötter-Vardy decoding procedures are widely perceived as the first algorithms capable of decoding RS codes beyond the *error-correction bound*, decoding architectures (non-specific to the type of code) were developed by Forney [8] in 1966 and Chase [9] in 1972 which utilize reliability information available at the channel output to incur a performance advantage over HDD procedures. Forney's *Generalized Minimum Distance* (GMD) architecture is characterized by successively applying *erasures-and-errors* decoding to a received vector as the set of erasures, initially taken as the the $(d_{\text{min}} - 1)$ least reliable locations, is reduced to size zero. Since any combination of $n_{err}$ errors and $n_{ers}$ erasures can be decoded correctly provided $n_{err}+n_{ers}/2 < d_{\text{min}}/2$, the error-correction capability of this procedure can be extended to nearly twice that of *errors-only* decoding. Efficient algorithms for GMD decoding (and approximations thereof) of RS codes

Manuscript was received April, 2006 and revised May, 2008. The material in the manuscript was presented, in part, at the International Symposium on Information Theory, Seattle, WA (July 9-14, 2006)

Jason Bellorado is with *Link-A-Media Devices*, Santa Clara, CA.

Aleksandar Kavčić is with the *Department of Electrical Engineering*, University of Hawaii, Honolulu, HI.

have been proposed ([10], [11] and [12]), which are further discussed in Sections IV and V.

In this work, however, we consider the decoding of RS code based on the architecture of Chase [9]. The Chase method, in which a multitude of hard-decision vectors comprising a *test-set* are individually applied as inputs to an existing HDD algorithm, can offer significant improvements over HDD for use in Reed-Solomon decoding. However, the high complexity required to conduct Chase-type decoding (which is linear in the test-set cardinality) for large test-sets has relegated its use primarily to applications such as *deep-sector recovery* for magnetic storage systems in which the need for performance outweighs its high complexity. In such scenarios, HDD may be conducted thousands of times on a given transmission and, therefore, application of Chase-type techniques for normal (online) decoding operation is often considered to be infeasible.

The complexity of Chase decoding, however, may be significantly reduced over this traditional approach (we note here that a similar approach as that presented was considered in [13] and [14]), which acts to decrease the average decoding complexity of Chase RS and BCH decoding. Here, we consider the maximum complexity required to decode, a widely utilized figure-of-merit for computational complexity. Since the test-set consists of the vectors of highest *a-posteriori probabilities*, they tend to be equivalent in a large number of coordinate positions. This is particularly true in the case of memoryless channels, in which the vectors of highest *a-posteriori probabilities* are obtained by substituting likely error events into the coordinate positions of lowest reliability (we explicitly show this in the following sections). In this work, we detail a procedure, building from the interpolation procedure presented in [15], which exploits this similarity among test-vectors to obtain the set of candidate codewords $\hat{\mathcal{C}}$ using only a fraction of the computation required to individually decode each vector (we note here that, although the efficiency of the *B-M* algorithm is widely known, it is not amenable to an exploitation of the similarity of the test-vectors since computation in the *B-M* algorithm is not conducted on the test-vectors, but on their syndromes, which are dissimilar even for similar test-vectors). Furthermore, we also present a reduced-complexity version of this algorithm which does not require an explicit computation of each element of $\hat{\mathcal{C}}$, but rather only interpolates the elements of the hard-decision test-vectors $\mathbf{Y}$ into a set of bivariate polynomials. A simple procedure is then presented to select a single polynomial which is likely to produce a correctly decoded codeword. We give results which show that significant gains over HDD are achievable with a computational complexity that is comparable to (and in some cases below that of) traditional approaches to HDD (such as Berlekamp-Massey). We also show that the procedure can produce performance similar to that of the Kötter-Vardy decoder [7], but at a much lower computational cost.

The remainder of this paper is structured as follows: In Section II we provide some background and notation used throughout this study. Here, we discuss coding, modulation, channel model, signal-to-noise ratio, and test-set construction. In Section III we describe our low-complexity Chase (LCC) decoding methodology, which is broken into three sections: test-set modification, polynomial interpolation, and polynomial factorization. In such, we provide both a method for full factorization and a reduced-complexity factorization (RCF) procedure. In Section IV we provide an analysis of the computational complexity of the presented algorithm and give a comparison to the complexity of HDD (using the *Berlekamp-Massey Algorithm*) and Kötter-Vardy (*K-V*) decoding. We provide a comparison of the performance of these methodologies in Section V.

## II. Background and Notation

### A. Coding and Modulation

In this work, we consider Reed-Solomon codes of length $n$, dimension $k$, and, due to their *Maximum Distance Separable* (MDS) nature [2], minimum distance $d_{\text{min}} = n - k + 1$. These codes are over the Galois Field of size $2^q$, which we denote as GF($2^q$), and, therefore, all algebraic operations are performed within this field. Although we acknowledge that a full definition of any RS code requires, in addition, the specification of its $(n-k)$ consecutive spectral zeros, all results presented hold irrespective of this distinction and, thus, it is omitted here. Any RS code of the given parameters, denoted by $\mathcal{C}_q(n,k)$, maps a $k$-tuple of information symbols $\mathbf{m} = (m_0, m_1, \ldots, m_{k-1}),\ m_i \in \text{GF}(2^q)$, to an $n$-tuple codeword $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1}),\ c_i \in \text{GF}(2^q)$, where $\mathbf{c} \in \mathcal{C}_q(n,k)$. Here, we consider an *evaluation-map* encoding of message symbols to codeword symbols, as given in the original definition of RS codes in [1]. In such, the message symbols are used as coefficients of a message polynomial $m(x)$ as,

$$m(x) = \sum_{i=0}^{k-1} m_i x^i \tag{1}$$

and the codeword is constructed as the evaluation of $m(x)$ at $n$ distinct elements of GF($2^q$). Thus, if we denote an ordered set of distinct elements of GF($2^q$) as $\mathcal{F} = \{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\}$, then the codeword is constructed as,

$$\mathbf{c} = (m(\alpha_0), m(\alpha_1), \ldots, m(\alpha_{n-1})). \tag{2}$$

Here, we note that our choice to consider an *evaluation-map* encoding is because it lends an insightful interpretation of the interpolation procedure that we use for decoding. However, the results we present hold for all encoding procedures (such as *systematic encoding*, *Galois Field Fourier Transform encoding*, and *generator polynomial encoding*) a comprehensive discussion of which may be found in [16].

The decoding methodology we present in this study is not dependent on the modulation employed or the channel under consideration. However, we do present simulation results which both motivate the achievable performance gains of Chase decoding over traditional HDD and verify the effectiveness of a suboptimal, reduced-complexity, factorization procedure detailed in this work. For these simulations we consider Binary Phase-Shift Keying (BPSK) modulation, in which a transmitted *binary* digit is mapped to the signal constellation $\mathcal{S} = \{-d, +d\}$, where $d$ is a parameter which controls the transmitted power. Since binary modulation is considered, we modulate each symbol of the codeword to a length-$q$ vector of constellation points through a bijection $\mathcal{M}$ : GF$(2^q) \rightarrow \mathcal{S}^q$. The transmitted vector is, thus, formed as $\mathbf{x} = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{n-1})$, where $\mathbf{x}_i = (x_{iq}, x_{iq+1}, \ldots, x_{(i+1)q-1}) = \mathcal{M}(c_i)$. We will use the notation $\mathcal{M}^{-1} : \mathcal{S}^q \rightarrow$ GF$(2^q)$ to denote the inverse of this bijection and, therefore, it must be true that $\mathcal{M}^{-1}(\mathbf{x}_i) = c_i$. We note that this bijection is introduced only to aid in the description of our algorithm. However, since it has no affect on the presented performance results, is not further discussed.

### B. Channel Model and Signal-to-Noise Ratio

The simulation results we present consider an additive white Gaussian noise (AWGN) channel model. For a single codeword, the length-$(nq)$ received vector $\mathbf{r}$ is given by the equation

$$\mathbf{r} = \mathbf{x} + \mathbf{n}. \tag{3}$$

Here, $\mathbf{x}$ is the length-$(nq)$ transmission vector of *binary* symbols from the signal set $\mathcal{S}$, mapped to by the codeword $\mathbf{c}$ as outlined in the preceding section. The length-$(nq)$ additive noise vector $\mathbf{n}$ is composed of independent, identically distributed, Gaussian random variables with zero mean and unit variance. Here, $\mathbf{n}$ is obtained by passing a white Gaussian noise process (which has a constant power spectral density (PSD) of magnitude $N_0/2$) through a bandlimited filter [17]. Normalizing by the signal bandwidth, the unit energy noise, thus, has a power spectrum defined by the parameter $N_0 = 2$.

We present our simulation results as a function of $E_b/N_0$, where $E_b$ is the transmitted energy per *information bit*. Each frame, consisting of $nq$ transmissions from the binary signal constellation $\mathcal{S}$, carries $kq$ information bits across the channel using $nqd^2$ Joules of energy. The energy transmitted per information bit is, thus, given by $E_b = d^2 n/k$, and

$$\frac{E_b}{N_0} = \frac{d^2 n}{2k}. \tag{4}$$

Aside from the fact that $E_b/N_0$ has, historically, been used for systems employing binary modulation, results presented as a function of this parameter may be easily compared, irregardless of code length, code rate, or modulation employed.

### C. Test-Set Construction

Our goal is to construct a test-set of hard-decision vectors which maximizes the probability that at least one test-vector will decode to the transmitted codeword when applied to an HDD algorithm. For an RS code, this is equivalent to maximizing the probability that each test-vector has no more than $(n-k+1)/2$ hard-decision *symbol-errors*. (Note that the term *symbol* is used to denote an element of GF$(2^q)$ and, thus, a symbol represents $q$ consecutive *binary* transmissions when *binary* modulation is used. A *symbol-error* occurs when any of its *binary* associated transmissions are in error.) Clearly, satisfying this criterion depends on the channel model being considered. For the AWGN channel given by (3), the probability that a hard-decision vector has less than $(n-k+1)/2$ symbol-errors may be obtained, closed-form, using a *binomial distribution*. Since this probability is maximized by maximizing the probability that each hard-decision made is correct, the single vector which has the highest probability of having less than $(n-k+1)/2$ symbol-errors is the *maximum a-posteriori* hard-decision vector (which we discuss in the following). The hard-decision vector of second highest probability of having less than $(n-k+1)/2$ symbol-errors may, subsequently, be obtained by changing the symbol of lowest reliability to its next most reliably value. This reasoning is followed through to construct our test-set.

Given the channel output vector $\mathbf{r}$, the probability that a vector $\mathbf{x} \in \mathcal{S}^{nq}$ was transmitted is given by the *a-posteriori probability*

$$p(\mathbf{x}|\mathbf{r}). \tag{5}$$

We denote by $\hat{\mathbf{x}}^{\text{(MAP)}}$ the single candidate transmission which maximizes the *a-posteriori probability*, i.e.,

$$\hat{\mathbf{x}}^{\text{(MAP)}} = \arg \max_{\mathbf{x} \in \mathcal{S}^{nq}} p(\mathbf{x}|\mathbf{r}). \tag{6}$$

Since the *Bayes' Law* may be used to rewrite the *a-posteriori probability* as

$$p(\mathbf{x}|\mathbf{r}) = \frac{p(\mathbf{r}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{r})}, \tag{7}$$

and because all *binary* symbols are, *a-priori*, equally probable, the MAP vector is equivalent to the *maximum-likelihood* (ML) likelihood vector, i.e.,

$$\hat{\mathbf{x}}^{\text{(MAP)}} = \arg \max_{\mathbf{x} \in \mathcal{S}^{nq}} p(\mathbf{r}|\mathbf{x}). \tag{8}$$

Due to the independence of the elements of the noise vector, $p(\mathbf{r}|\mathbf{x})$ may be factored as

$$p(\mathbf{r}|\mathbf{x}) = \prod_{i=0}^{nq-1} p(r_i|x_i). \tag{9}$$

Expressing $p(\mathbf{r}|\mathbf{x})$ in this form simplifies computation of MAP vector $\hat{\mathbf{x}}^{\text{(MAP)}} = (\hat{\mathbf{x}}_0^{\text{(MAP)}}, \hat{\mathbf{x}}_1^{\text{(MAP)}}, \ldots, \hat{\mathbf{x}}_{n-1}^{\text{(MAP)}})$. Thus we can individually maximize each term in (9) and hence $\hat{\mathbf{x}}^{\text{(MAP)}}$ may be be solved for on a symbol-by-symbol basis as,

$$\hat{\mathbf{x}}_i^{\text{(MAP)}} = \arg \max_{\mathbf{x}_i \in \mathcal{S}^q} p(\mathbf{x}_i|\mathbf{r}_i). \tag{10}$$

or a bit-by-bit basis as,

$$\hat{x}_i^{\text{(MAP)}} = \arg \max_{x_i \in \mathcal{S}} p(x_i|r_i), \tag{11}$$

where $\hat{\mathbf{x}}_i^{\text{(MAP)}} = (\hat{x}_{iq}^{\text{(MAP)}}, \hat{x}_{iq+1}^{\text{(MAP)}}, \ldots, \hat{x}_{(i+1)q-1}^{\text{(MAP)}})$. Utilizing the inverse bijection $\mathcal{M}^{-1}(\cdot)$, 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 $\hat{\mathbf{x}}^{\text{(MAP)}}$ as

$$y_i^{\text{HD}} = \mathcal{M}^{-1}(\hat{x}_{iq}^{\text{(MAP)}}, \hat{x}_{iq+1}^{\text{(MAP)}}, \ldots, \hat{x}_{(i+1)q-1}^{\text{(MAP)}}). \tag{12}$$

Because the vector $\mathbf{y}^{\text{HD}}$ is the element of GF$(2^q)^n$ of highest *a-posteriori probability*, it will, on average, minimize the number of *bit-errors* and, consequently, the number of *symbol-errors* among all vectors in GF$(2^q)^n$. Thus, the probability of an HDD error is minimized by the application of $\mathbf{y}^{\text{HD}}$ to the decoding algorithm.

We formalize the test-set construction by first defining the secondary MAP vector $\hat{\mathbf{x}}^{\text{(MAP2)}}$,

$$\hat{\mathbf{x}}^{\text{(MAP2)}} = (\hat{\mathbf{x}}_0^{\text{(MAP2)}}, \hat{\mathbf{x}}_1^{\text{(MAP2)}}, \dots, \hat{\mathbf{x}}_{n-1}^{\text{(MAP2)}}), \tag{13}$$

where each symbol $\hat{\mathbf{x}}_i^{\text{(MAP2)}}$ is the element of $\mathcal{S}^q$ that has the second largest *a-posteriori probability*, i.e.,

$$\hat{\mathbf{x}}_i^{\text{(MAP2)}} = \arg \max_{\mathbf{x}_i \in \mathcal{S}^q, \mathbf{x}_i \neq \hat{\mathbf{x}}_i^{\text{(MAP)}}} p(\mathbf{x}_i|\mathbf{r}_i). \tag{14}$$

Using $\hat{\mathbf{x}}^{\text{(MAP2)}}$, a secondary hard-decision vector $\mathbf{y}^{\text{2HD}} = (y_0^{\text{2HD}}, \dots, y_{n-1}^{\text{2HD}})$ is, then, given by,

$$y_i^{\text{2HD}} = \mathcal{M}^{-1}(\hat{\mathbf{x}}_i^{\text{(MAP2)}}). \tag{15}$$

Using $\mathbf{y}^{\text{HD}}$, $\mathbf{y}^{\text{2HD}}$, and $\mathbf{r}$ we construct the test-set by first computing a metric $\gamma_i$ for each coordinate location,

$$\gamma_i = \frac{p(c_i = y_i^{\text{2HD}}|\mathbf{r})}{p(c_i = y_i^{\text{HD}}|\mathbf{r})} \leq 1. \tag{16}$$

By definition of the hard-decisions and secondary hard-decisions, $\gamma_i$ is a real number in the interval $[0, 1]$, which gives a measure of the reliability of each coordinate location. Values of $\gamma_i$ close to 1 indicate a high probability that the hard-decision is in error and the secondary hard-decision is correct, while values of $\gamma_i$ close to zero indicate that the hard-decision is, with high probability, the transmitted value. Thus, for a given (a-priori chosen) positive integer $\eta$, we select the set of the lowest reliability indices as the coordinate positions with the $\eta$ largest values of $\gamma_i$. We will hereafter define $\mathcal{I} = \{i_1, i_2, \dots, i_\eta\}$ as this set of coordinate positions, and $\bar{\mathcal{I}} = \{\bar{i}_1, \bar{i}_2, \dots, \bar{i}_{n-\eta}\} = \{0, 1, \dots, n-1\} \setminus \mathcal{I}$ as the remaining positions. The test-set is, then, constructed as the set of all vectors $\mathbf{y}_i = (y_{i,0}, \dots, y_{i,n-1})$, with

$$y_{i,j} \in \begin{cases} \{y_j^{\text{HD}}\} & if \quad j \in \bar{\mathcal{I}} \\ \{y_j^{\text{HD}}, y_j^{\text{2HD}}\} & if \quad j \in \mathcal{I}. \end{cases} \tag{17}$$

Clearly, since we consider all vectors with both the hard-decision and secondary hard-decision in the indices in $\mathcal{I}$, the test-set, which we denote as $\mathbf{Y}$, has cardinality $2^{|\mathcal{I}|} = 2^\eta$.

## III. Low-Complexity Chase (LCC) Decoding

In this section, we present a methodology to decode each test-vector $\mathbf{y}_i$ into the message polynomial $\hat{m}_i(x)$ with an *evaluation-map* codeword $\hat{\mathbf{c}}_i \in \mathcal{C}_q(n, k)$ of Hamming distance no greater than $(n-k+1)/2$ from $\mathbf{y}_i$ (provided such a message exists). The procedure we detail first interpolates the test-vector $\mathbf{y}_i$ into a bivariate polynomial $Q_i(x, z)$, which contains $\hat{m}_i(x)$ in its, single, $z$-linear factor. In doing so, we exploit the high degree of similarity among the test-vectors (see (17)) to achieve a large reduction in the complexity required for this decoding step. We, further, present an expression for the explicit computation of the $z$-linear factor of $Q_i(x, z)$, and the candidate message polynomial contained therein. This calculation, however, must be conducted for all $2^\eta$ interpolation polynomials. Motivated by the high complexity ($O(2^\eta)$) of this required repetition, as well as the complexity of re-encoding each $\hat{m}_i(x)$ into $\hat{\mathbf{c}}_i$ such that a comparison of *a-posteriori probabilities* can be made, we also present a reduced-complexity algorithm. Here, a minimal amount of computation on each $Q_i(x, z)$ is used to select a single polynomial $Q^*(x, z)$ which is, subsequently, factored to obtain the candidate message polynomial. Although suboptimal, we provide simulation results that show the convergence of its performance to explicitly factoring each polynomial with increasing code length $n$.

### A. Test-Set Modification

We first present a method to modify the elements of the test-set which allows for a significant reduction in the computation required to interpolate the test-vectors, a step that was first presented as *re-encoding* in [15]. Although this step is not required, the overall decoding complexity is reduced by its inclusion.

Any test-vector $\mathbf{y} = (y_0, y_1, \dots, y_{n-1}) \in \mathbf{Y}$ may be written as the sum of the transmitted codeword $\mathbf{c}$ and an error-vector $\mathbf{e}$ as

$$\mathbf{y} = \mathbf{c} + \mathbf{e}. \tag{18}$$

Denoting any set of $k$ coordinate positions by $\mathcal{J} = \{j_1, j_2, \dots, j_k\} \subset \{0, 1, \dots, n-1\}$, we define a codeword $\Psi = (\psi_0, \psi_1, \dots, \psi_{n-1}) \in \mathcal{C}_q(n, k)$ which we constrain in the coordinate locations in $\mathcal{J}$ as,

$$\psi_i = y_i \quad \text{for every } i \in \mathcal{J}. \tag{19}$$

The existence of a codeword $\Psi$ satisfying (19) is guaranteed by the fact that RS codes are capable of correcting up to $d_{\text{min}} - 1 = n - k$ erasures [2]. Thus, determining $\Psi$ is equivalent to an erasures-only decoding of $\mathbf{y}$, where the coordinate locations in $\bar{\mathcal{J}} = \{0, 1, \dots, n-1\} \setminus \mathcal{J}$ are erased. Adding $\Psi$ to $\mathbf{y}$ yields,

$$\begin{aligned} \mathbf{y}' &= \mathbf{y} + \Psi \\ &= \mathbf{c} + \Psi + \mathbf{e} \\ &= \mathbf{c}' + \mathbf{e}, \end{aligned} \tag{20}$$

where $\mathbf{c}' = \mathbf{c} + \Psi \in \mathcal{C}_q(n, k)$ due to the linearity of RS codes. Furthermore, since we have added a codeword to $\mathbf{y}$ to obtain $\mathbf{y}'$, they are both in the same coset (have the same error vector). This fact is important because obtaining a codeword from a hard-decision vector is equivalent to obtaining its error-vector and, thus, any member of a given coset may be used for decoding. In this case, we utilize the coset member of minimum Hamming weight $= (n - k) = d_{\text{min}}$. From the previous discussion, it is clear that every coset contains such a codeword.

We have, thus far, presented how a single test-vector $\mathbf{y}$ is modified to obtain a vector $\mathbf{y}'$ of minimum Hamming weight in the same coset. However, since we intend to exploit the similarity among test-vectors in $n - \eta$ locations corresponding to the elements of $\bar{\mathcal{I}}$, we must ensure that the modification procedure does not adversely affect this attribute (i.e. we desire the modified test-vectors to also be equivalent in these $n - \eta$ coordinate locations). Since the vector $\Psi$ is constructed using an erasures-only decoding with the coordinate locations in $\bar{\mathcal{J}}$ erased, $\Psi$ depends only on the value of $\mathbf{y}$ in the coordinate locations in $\mathcal{J}$. We, therefore, select $\mathcal{J}$ such that,

$$\mathcal{J} \subseteq \bar{\mathcal{I}}, \tag{21}$$

since

$$y_{i,j} = y_j^{\text{HD}}, \quad \text{for every } (i,j) \in \{1,2,\ldots,2^\eta\} \times \bar{\mathcal{I}}. \qquad (22)$$

(We note here that, in selecting $\mathcal{J}$ according to (21), we have tacitly assumed that $\eta \leq (n-k)$. Although we are imposing this restriction, the complexity results we present in Section IV show that a (computationally) practical selection of $\eta$ is upper bounded by $\log_2(n-k)+1$. Thus, since the decoding complexity for values of $\eta$ larger than $(n-k)$ is prohibitive, we are not unduely limited by this assumption.) By selecting $\mathcal{J}$ as any set of indices satisfying (21), we force $\Psi$ to be equivalent for all test-vectors. We modify each test-vector as,

$$\mathbf{y}'_i = \mathbf{y}_i + \Psi, \qquad (23)$$

the set of which we denote by $\mathbf{Y}'$, and maintain their equivalence on the coordinate locations in $\bar{\mathcal{I}}$. At this point, we remark that any choice of $\mathcal{J}$ satisfying (21) will suffice, however, we will require a more judicious choice to accomplish the reduced-complexity factorization procedure detailed in Section III-C2.

## B. Polynomial Interpolation

To obtain a candidate message polynomial $\hat{m}_i(x)$ from each of the modified test-vectors $\mathbf{y}'_i \in \mathbf{Y}'$, we interpolate $\mathbf{y}'_i$ into a bivariate polynomial $Q_i(x,z)$ which contains $\hat{m}_i(x)$ in its $z$-linear factor. To prove that this is, indeed, the case, we first make the following definition:

*Definition 1*: The $(1,m)$-weighted degree of a bivariate polynomial $Q(x,z) = \sum_i \sum_j q_{i,j} x^i z^j$, as denoted by $\deg_{(1,m)}(Q(x,z))$, is defined as,

$$\deg_{(1,m)}(Q(x,z)) = \max\,(i + m \cdot j) \qquad (24)$$

where the maximization is taken over all values $i,j$ such that $q_{i,j}x^iz^j$ is a monomial in $Q(x,z)$ and $q_{i,j} \neq 0$. □

Using *Definition 1*, our assertion that a $z$-linear factor of $Q_i(x,z)$ contains the message polynomial decoded from vector $\mathbf{y}_i$ is validated by *Theorem 1*.

*Theorem 1*: Consider a hard-decision vector $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1})$ and the bivariate polynomial $Q(x,z)$ of minimum $(1,k-1)$-weighted degree for which the following holds,

$$Q(x,z) = 0 \;\; \text{for } (x,z) \in \{(\alpha_0,y_0),\ldots,(\alpha_{n\text{-}1},y_{n\text{-}1})\} \quad (25)$$

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 [6] for multiplicity = 1. ■

We also have the following *Corollary* to *Theorem 1*.

*Corollary 1*: The bivariate polynomial $Q(x,z)$ specified by *Theorem 1* has $z$-degree of at most one. □
*Proof*: See [6]. ■

To accomplish the test-vector interpolation, we utilize an algorithm presented in [15] (this algorithm, commonly referred to as *Nielsen's Algorithm* [18], is a modified version of the interpolation algorithm presented in [7]). The proposed algorithm interpolates the vector $\mathbf{y}'_i = (y'_{i,0}, y'_{i,1}, \ldots, y'_{i,n-1})$ by considering each coordinate position individually. If we denote an arbitrary set of $m$ coordinate positions by $\mathcal{P}^{(m)} \subseteq \{0,1,\ldots,n-1\}$, the procedure, at the completion of the $m^{th}$ interpolation step, constructs an intermediate bivariate polynomial $Q_i^{(m)}(x,z)$ which satisfies the constraints,

$$Q_i^{(m)}(\alpha_j, y'_{i,j}) = 0 \;\; \text{for every } j \in \mathcal{P}^{(m)}. \qquad (26)$$

At the completion of the $n^{th}$ interpolation step, the polynomial $Q_i^{(n)}(x,z)$ has roots at all pairs $(\alpha,\beta) \in \{(\alpha_0,y'_{i,0}),(\alpha_1,y'_{i,1}),\ldots,(\alpha_{n-1},y'_{i,n-1})\}$ and, therefore,

$$Q_i^{(n)}(x,z) = Q_i(x,z). \qquad (27)$$

It is important to note that (27) holds regardless of the order in which the coordinate locations are considered and, thus, we are free to choose this ordering.

As shown in [15], the coordinate positions in $\mathcal{J}$ are easily interpolated by exploiting the fact that all test-vectors are zero in these indices. This is accomplished by constructing the polynomial $v(x)$,

$$v(x) = \prod_{i \in \mathcal{J}} (x + \alpha_i), \qquad (28)$$

and making use of the change of variables,

$$\tilde{z} = \frac{z}{v(x)}. \qquad (29)$$

As such, only the final $(n-k)$ coordinate positions in $\bar{\mathcal{J}}$ require explicit interpolation, while the indices in $\mathcal{J}$ are accounted for at the completion of interpolation. As a direct result of the change of variables, the interpolation pairs are first translated accordingly, i.e.,

$$(\alpha_i, y'_i) \rightarrow (\alpha_i, \frac{y'_i}{v(\alpha_i)}), \qquad (30)$$

and the minimal $(1,k-1)$-weighted degree polynomial specified by *Theorem 1* becomes the minimal $(1,-1)$-weighted degree polynomial (see [15] or [19] for a complete treatment of this fact).

We conduct the interpolation of these $(n-k)$ indices by partitioning $\bar{\mathcal{J}}$ into the sets,

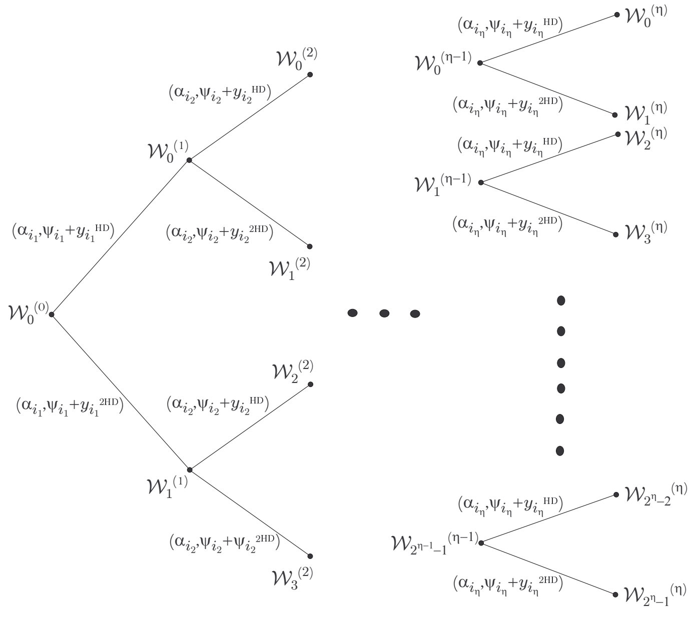
- $\mathcal{B}_c = \bar{\mathcal{J}} \cap \bar{\mathcal{I}}$
- $\mathcal{B}_u = \bar{\mathcal{J}} \cap \mathcal{I}$.

We first consider the coordinate positions in $\mathcal{B}_c$, of which there are $(n-k-\eta)$. Because the elements of $\mathcal{B}_c$ are also elements of $\bar{\mathcal{I}}$, the values of all test-vectors in these coordinate positions are equivalent, and of the form

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

**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
  Initialize $\mathcal{G} = \{1, z\}$
  For all $i \in \mathcal{B}_c$, compute $v_i = v(\alpha_i)$
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
  $\mathcal{G}_c = \{(g_0(x,z), g_1(x,z))\}$
```



$\mathcal{W}_0^{(0)}$
$\mathcal{W}_0^{(1)}$
$\mathcal{W}_1^{(1)}$
$\mathcal{W}_0^{(2)}$
$\mathcal{W}_1^{(2)}$
$\mathcal{W}_2^{(2)}$
$\mathcal{W}_3^{(2)}$
$(\alpha_{i_1}, \psi_{i_1} + y_{i_1}^{\text{HD}})$
$(\alpha_{i_1}, \psi_{i_1} + y_{i_1}^{\text{2HD}})$
$(\alpha_{i_2}, \psi_{i_2} + y_{i_2}^{\text{HD}})$
$(\alpha_{i_2}, \psi_{i_2} + y_{i_2}^{\text{2HD}})$
$(\alpha_{i_2}, \psi_{i_2} + y_{i_2}^{\text{HD}})$
$(\alpha_{i_2}, \psi_{i_2} + \psi_{i_2}^{\text{2HD}})$
$\mathcal{W}_0^{(\eta-1)}$
$\mathcal{W}_1^{(\eta-1)}$
$\mathcal{W}_{2^{\eta-1}-1}^{(\eta-1)}$
$\mathcal{W}_0^{(\eta)}$
$\mathcal{W}_1^{(\eta)}$
$\mathcal{W}_2^{(\eta)}$
$\mathcal{W}_3^{(\eta)}$
$\mathcal{W}_{2^\eta-2}^{(\eta)}$
$\mathcal{W}_{2^\eta-1}^{(\eta)}$
$(\alpha_{i_\eta}, \psi_{i_\eta} + y_{i_\eta}^{\text{HD}})$
$(\alpha_{i_\eta}, \psi_{i_\eta} + y_{i_\eta}^{\text{2HD}})$

Fig. 1. Graphical representation of the *Uncommon-Element Interpolation* algorithm. Each vertex represents a pair of polynomials interpolated from all branch labels leading from the root of the tree to that vertex.

Thus, interpolation of these coordinate positions takes the form of the interpolation of a single test-vector and, thus, application of existing techniques is possible [15]. We provide this procedure as **Algorithm 1**, *Common-Element Interpolation*.

The final $\eta$ coordinate locations not yet interpolated are from the set $\mathcal{B}_u$, the *uncommon element set*. Although the test-vectors are not equivalent in this set of coordinate positions, they take on one of the following two forms

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

We, therefore, must interpolate data-points taking on both of these forms. Since we seek the minimum $(1,-1)$-weighted degree polynomial (due to the change of variables) with roots specified by each test-vector for *Theorem 1* to apply, we cannot simply interpolate both points into the same polynomial. Each data-point must be interpolated into a separate polynomial, thus, requiring an expansion of the polynomial set. The end result is a set of polynomials, with each polynomial corresponding to a test-vector in $\mathbf{Y}'$.

We begin the *Uncommon-Element Interpolation Algorithm* with the set $\mathcal{G}_0 = \{(g_0(x,z), g_1(x,z))\}$ containing the single pair of polynomials constructed from **Algorithm 1**. Making the definition,

$$\mathcal{W}_0^{(0)} \triangleq (g_0(x,z), g_1(x,z)) \tag{31}$$

we rewrite $\mathcal{G}_0$ as

$$\mathcal{G}_0 = \{\mathcal{W}_0^{(0)}\}. \tag{32}$$

Although we are free to interpolate the indices $i_1, i_2, \ldots, i_\eta$ in any order, we utilize the given ordering in our presentation of the algorithm. With this assumption, our intention is, after the interpolation of the first $l$ indices $i_1, i_2, \ldots, i_l$, to have constructed the set $\mathcal{G}_l$,

$$\mathcal{G}_l = \{\mathcal{W}_0^{(l)}, \mathcal{W}_1^{(l)}, \ldots, \mathcal{W}_{2^l-1}^{(l)}\}, \tag{33}$$

where $\mathcal{W}_i^{(l)}$ is the pair of polynomials given by

$$\mathcal{W}_i^{(l)} = (g_{i,0}^{(l)}(x,z), g_{i,1}^{(l)}(x,z)). \tag{34}$$

Denoting the $l$-bit binary representation of $i$ by $(b_{i,1}b_{i,2}\ldots b_{i,l})$, the roots of $g_{i,0}^{(l)}(x,z)$ and $g_{i,1}^{(l)}(x,z)$ are given, for $j \in \{1,2,\ldots,l\}$, by

$$g_{i,0}^{(l)}(\alpha_{i_j}, \psi_{i_j} + y_{i_j}^{\text{HD}}) = g_{i,1}^{(l)}(\alpha_{i_j}, \psi_{i_j} + y_{i_j}^{\text{HD}}) = 0, \text{ iff } b_{i,j} = 0 \tag{35}$$

$$g_{i,0}^{(l)}(\alpha_{i_j}, \psi_{i_j} + y_{i_j}^{\text{2HD}}) = g_{i,1}^{(l)}(\alpha_{i_j}, \psi_{i_j} + y_{i_j}^{\text{2HD}}) = 0, \text{ iff } b_{i,j} = 1. \tag{36}$$

Furthermore, because all polynomials are constructed starting from $\mathcal{W}_0^{(0)}$, it is also true that, for all $j \in \mathcal{B}_c$,

$$g_{0,i}^{(l)}(\alpha_j, \psi_j + y_j^{\text{HD}}) = g_{1,i}^{(l)}(\alpha_j, \psi_j + y_j^{\text{HD}}) = 0. \tag{37}$$

It, therefore, follows that the set $\mathcal{G}_\eta$ contains $2^\eta$ polynomial pairs, each of which being interpolated from a single test-vector in $\mathbf{Y}'$.

The procedure for the construction of $\mathcal{G}_\eta$ is depicted graphically as a binary tree in Figure 1. Here, the interpolation of coordinate position $i_l$ is represented as a progression from a depth $(l-1)$ to a depth $l$ in the tree. In such, we construct the set of polynomials specified by the depth-$l$ nodes $\mathcal{G}_l = \{\mathcal{W}_0^{(l)}, \mathcal{W}_1^{(l)}, \ldots, \mathcal{W}_{2^l-1}^{(l)}\}$ from the polynomials specified by the depth-$(l-1)$ nodes $\mathcal{G}_{l-1} = \{\mathcal{W}_0^{(l-1)}, \mathcal{W}_1^{(l-1)}, \ldots, \mathcal{W}_{2^{l-1}-1}^{(l-1)}\}$ by interpolating the data-points labeling each connecting branch. As can be seen, producing polynomial pairs $\mathcal{W}_{2p}^{(l)}$ and $\mathcal{W}_{2p+1}^{(l)}$ is accomplished by interpolating polynomial pair $\mathcal{W}_p^{(l-1)}$ through data-points

$(\alpha_{i_{l+1}}, \psi_{i_{l+1}} + y^{\text{HD}}_{i_{l+1}})$ and $(\alpha_{i_{l+1}}, \psi_{i_{l+1}} + y^{\text{2HD}}_{i_{l+1}})$, respectively. This interpolation may be conducted by individually applying a single step of **Algorithm 1** for each of these data-points to the polynomial pair $\mathcal{W}_p^{(l-1)}$. The computation required for this approach, which is dominated by the *compute* and *update* steps, is doubled over the interpolation of a single data-point.

We significantly reduce this computation by interpolating both data-points jointly and exploiting the fact that a common $x$-value is shared by these points. For this, we first note that, by *Corollary 1*, the $z$-degree of any interpolation polynomial can not exceed one and, therefore, we express $\mathcal{W}_i^{(l-1)} = (g_{i,0}^{(l-1)}(x,z), g_{i,1}^{(l-1)}(x,z))$ as

$$g_{i,0}^{(l-1)}(x,z) = g_{0,0}(x) + z \cdot g_{0,1}(x) \quad (38)$$

$$g_{i,1}^{(l-1)}(x,z) = g_{1,0}(x) + z \cdot g_{1,1}(x). \quad (39)$$

Since we are first required to evaluate these polynomials at both data-points (see **Algorithm 1**), we first compute the values

$$c_{0,0} = g_{0,0}(\alpha_{i_{l+1}}), \quad c_{0,1} = g_{0,1}(\alpha_{i_{l+1}}) \quad (40)$$

$$c_{1,0} = g_{1,0}(\alpha_{i_{l+1}}), \quad c_{1,1} = g_{1,1}(\alpha_{i_{l+1}}). \quad (41)$$

and compute the polynomial evaluations as,

$$g_{i,0}^{(l-1)}(\alpha_{i_{l+1}}, \psi_{i_{l+1}} + y^{\text{HD}}_{i_{l+1}}) = c_{0,0} + (\psi_{i_{l+1}} + y^{\text{HD}}_{i_{l+1}}) \cdot c_{0,1} \quad (42)$$

$$g_{i,1}^{(l-1)}(\alpha_{i_{l+1}}, \psi_{i_{l+1}} + y^{\text{HD}}_{i_{l+1}}) = c_{1,0} + (\psi_{i_{l+1}} + y^{\text{HD}}_{i_{l+1}}) \cdot c_{1,1} \quad (43)$$

$$g_{i,0}^{(l-1)}(\alpha_{i_{l+1}}, \psi_{i_{l+1}} + y^{\text{2HD}}_{i_{l+1}}) = c_{0,0} + (\psi_{i_{l+1}} + y^{\text{2HD}}_{i_{l+1}}) \cdot c_{0,1} \quad (44)$$

$$g_{i,1}^{(l-1)}(\alpha_{i_{l+1}}, \psi_{i_{l+1}} + y^{\text{2HD}}_{i_{l+1}}) = c_{1,0} + (\psi_{i_{l+1}} + y^{\text{2HD}}_{i_{l+1}}) \cdot c_{1,1}. \quad (45)$$

By carrying out the polynomial evaluations in this manner, only two additional multiplications are required over the evaluation at a single data-point.

The computation may also be reduced for the other main step of the interpolation procedure, the *update step*. To demonstrate this, without loss of generality, we will assume that

$$\deg_{(1,-1)}(g_{i,0}^{(l-1)}(x,z)) \leq \deg_{(1,-1)}(g_{i,1}^{(l-1)}(x,z)) \quad (46)$$

and consider three distinct cases.

First, we consider the case that neither $(\alpha_{i_{l+1}}, \psi_{i_{l+1}} + y^{\text{HD}}_{i_{l+1}})$ nor $(\alpha_{i_{l+1}}, \psi_{i_{l+1}} + y^{\text{2HD}}_{i_{l+1}})$ is a root of $g_{i,0}^{(l-1)}(x,z)$. By the rules of **Algorithm 1** we will choose

$$f_{\text{HD}}(x,z) = f_{\text{2HD}}(x,z) = g_{i,0}^{(l-1)}(x,z), \quad (47)$$

where $f_{\text{HD}}(x,z)$ and $f_{\text{2HD}}(x,z)$ denote the polynomial $f(x,z)$ for data-points $(\alpha_{i_{l+1}}, \psi_{i_{l+1}} + y^{\text{HD}}_{i_{l+1}})$ and $(\alpha_{i_{l+1}}, \psi_{i_{l+1}} + y^{\text{2HD}}_{i_{l+1}})$, respectively. Because the update of $f(x,z)$ depends only on the $x$-value of the data-point, i.e.,

$$f(x,z) = (x + \alpha_{i_{l+1}}) \cdot f(x,z) \quad (48)$$

we need only perform this update once, and perform both updates for the remaining polynomial $g(x,z) = \mathcal{G} \setminus f(x,z)$.

The second case we consider is when exactly one of the data-points is a root of $g_{i,0}^{(l-1)}(x,z)$. Without loss of generality, we assume that $g_{i,0}^{(l-1)}(\alpha_{i_{l+1}}, \psi_{i_{l+1}} + y^{\text{HD}}_{i_{l+1}}) = 0$ and, therefore, we choose

$$f_{\text{HD}}(x,z) = g_{i,1}^{(l-1)}(x,z) \quad (49)$$

$$f_{\text{2HD}}(x,z) = g_{i,0}^{(l-1)}(x,z). \quad (50)$$

Here, the dissimilar selection of $f(x,z)$ for the data-points necessitates both updates of this polynomial to be conducted. However, the update of $g_{\text{HD}}(x,z)$,

$$g_{\text{HD}}(x,z) = g_{\text{HD}}(x,z) + \left( \frac{g_{\text{HD}}(\alpha_i, \psi_{i_{l+1}} + y^{\text{HD}}_{i_{l+1}})}{f_{\text{HD}}(\alpha_i, \psi_{i_{l+1}} + y^{\text{HD}}_{i_{l+1}})} \right) f_{\text{HD}}(x,z) \quad (51)$$

is not required since $g_{\text{HD}}(\alpha_i, \psi_{i_{l+1}} + y^{\text{HD}}_{i_{l+1}}) = 0$. Thus, as was the situation in the previous case, only three of the four total updates are required.

We, finally, consider the third case in which both data-points are roots of $g_{i,0}^{(l-1)}(x,z)$, forcing the selection

$$f_{\text{HD}}(x,z) = f_{\text{2HD}}(x,z) = g_{i,1}^{(l-1)}(x,z). \quad (52)$$

Here, the equivalence of $f(x,z)$ among the data-points necessitates only the single update,

$$f_{\text{HD}}(x,z) = f_{\text{2HD}}(x,z) = (x + \alpha_{i_{l+1}}) \cdot f(x,z), \quad (53)$$

while the fact that both data-points are already roots of $g_{\text{HD}}(x,z) = g_{\text{2HD}}(x,z)$ removes the requirement for their updates.

The purpose of this illustration is to show that the computation required to produce the polynomial pairs $\mathcal{W}_{2p}^{(l)}$ and $\mathcal{W}_{2p+1}^{(l)}$ from $\mathcal{W}_p^{(l-1)}$ is only slightly higher than the interpolation of a single data-point (i.e., a single step of **Algorithm 1**). The complexity of the full algorithm used to produce the final set of polynomial pairs $\mathcal{G}_\eta$ (**Algorithm 2**) is provided in Section IV.

## *C. Polynomial Factorization*

The set $\mathcal{G}_\eta$, as constructed using **Algorithm 2**, contains pairs of polynomials $\{\mathcal{W}_0^{(\eta)}, \mathcal{W}_1^{(\eta)}, \ldots, \mathcal{W}_{2^\eta}^{(\eta)}\}$, where $\mathcal{W}_i^{(\eta)}$ is interpolated from the values of test-vector $\mathbf{y}'_i$ on the coordinate positions in $\bar{\mathcal{J}}$. As described in Section III-B, we require, for each test-vector, the single interpolation polynomial of lowest $(1,-1)$-weighted degree. We, therefore, define the set $\tilde{\mathcal{Q}} = \{\tilde{Q}_0(x,z), \tilde{Q}_1(x,z), \ldots, \tilde{Q}_{2^\eta - 1}(x,z)\}$, where

$$\tilde{Q}_i(x,z) \triangleq \arg \min_{a(x,z) \in W_i^{(\eta)}} \deg_{(1,-1)}(a(x,z)) \quad (54)$$

$$= \tilde{q}_{i,0}(x) + z \cdot \tilde{q}_{i,1}(x). \quad (55)$$

We, then, construct the minimal $(1, k-1)$-weighted degree polynomials interpolated from all coordinate positions, as required for the application of *Theorem 1*, by reverting back from the change of variables defined by (29) to obtain $\mathcal{Q} = \{Q_0(x,z), Q_1(x,z), \ldots, Q_{2^\eta}(x,z)\}$ where,

$$Q_i(x,z) = \tilde{q}_{i,0}(x) \cdot v(x) + z \cdot \tilde{q}_{i,1}(x). \quad (56)$$

We may directly apply *Theorem 1* by explicitly computing the $z$-linear factor of each $Q_i(x,z)$ and the candidate message polynomial contained therein. In doing so, we are

**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
  Initialize $\mathcal{G}_0$ as the set $\mathcal{G}_0 = \{\mathcal{W}_0^{(0)}\} = \mathcal{G}_c$
  Compute, for all $j \in \{1, 2, \ldots, \eta\}$, $v_{i_j} = v(\alpha_{i_j})$
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:
        $f_{\text{HD}}(x,z) = \arg\min_{g(x,z)\in\mathcal{G}'} \{\deg_{(1,-1)}(g(x,z)), g(\alpha_i, \tilde{y}_i^{\text{HD}}) \neq 0\}$
        $f_{\text{2HD}}(x,z) = \arg\min_{g(x,z)\in\mathcal{G}'} \{\deg_{(1,-1)}(g(x,z)), g(\alpha_i, \tilde{y}_i^{\text{2HD}}) \neq 0\}$
        $g_{\text{HD}}(x,z) = \mathcal{G}' \setminus \{f_{\text{HD}}(x,z)\}$
        $g_{\text{2HD}}(x,z) = \mathcal{G}' \setminus \{f_{\text{2HD}}(x,z)\}$
      update*:
        $g_{\text{HD}}(x,z)$ += $(g_{\text{HD}}(\alpha_i, \tilde{y}_i^{\text{HD}})/f_{\text{HD}}(\alpha_i, \tilde{y}_i^{\text{HD}})) \cdot f_{\text{HD}}(x,z)$
        $g_{\text{2HD}}(x,z)$ += $(g_{\text{2HD}}(\alpha_i, \tilde{y}_i^{\text{2HD}})/f_{\text{2HD}}(\alpha_i, \tilde{y}_i^{\text{2HD}})) \cdot f_{\text{2HD}}(x,z)$
        $f_{\text{HD}}(x,z) = (x + \alpha_i) \cdot f_{\text{HD}}(x,z)$
        $f_{\text{2HD}}(x,z) = (x + \alpha_i) \cdot f_{\text{2HD}}(x,z)$
        $\mathcal{W}_{2m}^{(j)} = (g_{\text{HD}}(x,z), f_{\text{HD}}(x,z))$
        $\mathcal{W}_{2m+1}^{(j)} = (g_{\text{2HD}}(x,z), f_{\text{2HD}}(x,z))$
    END FOR
  END FOR
OUTPUT
  $\mathcal{G}_\eta = \{\mathcal{W}_0^{(\eta)}, \mathcal{W}_1^{(\eta)}, \ldots, \mathcal{W}_{2^\eta-1}^{(\eta)}\}$
```

* Computation should be conducted as described in Section III-B.

left with a set of $2^\eta$ candidate message polynomials $\hat{\mathcal{M}} = \{\hat{m}_0(x), \hat{m}_1(x), \ldots, \hat{m}_{2^\eta-1}(x)\}$, from which a single decoded message $\hat{m}^*(x)$ must be chosen. To minimize the probability of decoding failure, we choose $\hat{m}^*(x)$ as,

$$\hat{m}^*(x) = \arg\max_{m(x)\in\hat{\mathcal{M}}} p(m(x)|\mathbf{r}). \tag{57}$$

We note here that, in order to compute (57), we need to encode and modulate each candidate message into a candidate transmission, such that (57) may be expressed in terms of Gaussian probability density functions. In the following section, we outline the methodology to obtain the set $\hat{\mathcal{M}}$ using an explicit factorization of each polynomial in $\mathcal{Q}$.

*1) Full Polynomial Factorization:* Because each interpolation polynomial $Q_i(x,z)$ has a $z$-degree that is no greater than one (see *Corollary 1*), its $z$-linear factor may be computed without the need for a root-finding algorithm (such as that presented in [20]). To show this, we note that because $(z - \hat{m}_i(x))$ divides $Q_i(x,z)$, the *Fundamental Theorem of Algebra* requires that

$$Q_i(x, \hat{m}_i(x)) = 0. \tag{58}$$

Substitution into (56) yields

$$\tilde{q}_{i,0}(x) \cdot v(x) + \hat{m}_i(x) \cdot \tilde{q}_{i,1}(x) = 0, \tag{59}$$

from which we obtain $\hat{m}_i(x)$ as,

$$\hat{m}_i(x) = \frac{\tilde{q}_{i,0}(x)v(x)}{\tilde{q}_{i,1}(x)}. \tag{60}$$

Applying (60) to each polynomial $Q_i(x,z)$, we produce the set of candidate message polynomials $\hat{\mathcal{M}}$. The decoded message polynomial $\hat{m}^*(x)$ is, then, chosen as the element of $\hat{\mathcal{M}}$ with highest *a-posteriori probability*.

The computation required for this approach, however, can be significant, particularly for use in decoding high-rate codes. We were able make use of a change-of-variables during the interpolation procedure to obviate the need to carry the degree-$k$ factor $v(x)$ through our computations. Interpolation was, thus, performed using the, much lower degree, polynomials in $\mathcal{G}_i, i = 1, 2, \ldots, \eta$ (an upper bound on the degrees of these polynomials is given in [19]). However, since $v(x)$ must be reintroduced for factorization, much of the computational savings achieved during interpolation are lost with a direct application of (60) to all elements of $\mathcal{Q}$.

*2) Reduced-Complexity Factorization (RCF):* Much of the complexity incurred by the full-factorization procedure presented in the previous section results from the need to conduct it for each of the $2^\eta$ interpolation polynomials. Furthermore, we must re-encode each candidate message polynomial such that a comparison of *a-posteriori probabilities* can be made. Here, we present a lower-complexity alternative to full-factorization, which we refer to as *Reduced-Complexity Factorization* (RCF). In such, we select a single interpolation polynomial that, with high probability, can be factored to obtain a correctly decoded message. We do note, however, that the full-factorization procedure is optimal with respect to the minimization of frame-error probability. Thus, since RCF only provides an approximation of full-factorization, its error-rate is lower bounded by that achieved with full-factorization. Although we provide no proof of the performance of RCF, we do provide simulation results in Section V to attest to its effectiveness for decoding long, high-rate, RS codes.

Before presenting our RCF procedure, we first make a small modification to the selection of the indices in $\mathcal{J}$. We had previously specified that $\mathcal{J}$ can be chosen as any, cardinality $k$, subset of $\bar{\mathcal{I}}$. However, we now choose $\mathcal{J}$ as the set of $k$ coordinate positions for which $\gamma_i$ (see (16)) is minimized. This problem is equivalently stated as choosing the set $\bar{\mathcal{J}}$ to contain the $(n-k)$ coordinate positions for which $\gamma_i$ is maximized. We, therefore, now require the determination of the $(n-k)$ least-reliable coordinate positions, whereas only $\eta$ were required in our previous definition of $\mathcal{J}$.

With this modification in place, we examine the expression given in (60) for the evaluation of each candidate message polynomial. Although $\hat{m}_i(x)$ is expressed as a quotient, it must be a valid message polynomial and, therefore, have a degree no greater than $(k-1)$. Thus, all irreducible factors of the denominator $\tilde{q}_{i,1}(x)$ must also be factors of the numerator

$\tilde{q}_{i,0}(x)v(x)$. If this were not the case, the degree of $\hat{m}_i(x)$ would be infinite, rendering it invalid.

At this point we make the assumption that $\tilde{q}_{i,1}(x)$ has no irreducible factors of degree larger than one. We remark that this need not be the case, however, any higher degree irreducible factor of $\tilde{q}_{i,1}(x)$ must also be a factor of $\tilde{q}_{i,0}(x)$, since $v(x)$ can have no such factors. We have found, empirically, that this assumption holds the vast majority of the time, provided $\hat{m}_i(x)$ is a valid message polynomial. Using this assumption, we express $\tilde{q}_{i,1}(x)$ as

$$\tilde{q}_{i,1}(x) = \zeta \cdot \prod_{\alpha \in \mathcal{A}_i} (x+\alpha) \cdot \prod_{\beta \in \mathcal{B}_i} (x+\beta), \qquad (61)$$

where $\zeta \in \text{GF}(2^q)$ is a scaling constant and $\mathcal{A}_i$ and $\mathcal{B}_i$ are the sets of roots that $\tilde{q}_{i,1}(x)$ shares with $v(x)$ and $\tilde{q}_{i,0}(x)$, respectively. Since the roots of $v(x)$ are known to be all elements of the set $\mathcal{R} = \{\alpha_{j_1}, \alpha_{j_2}, \ldots, \alpha_{j_k}\}$, any root of $\tilde{q}_{i,1}(x)$ in $\bar{\mathcal{R}} = \{\alpha_0, \alpha_1, \ldots, \alpha_{2^q-2}\} \backslash \{\alpha_{j_1}, \alpha_{j_2}, \ldots, \alpha_{j_k}\}$ must be shared with $\tilde{q}_{i,0}(x)$ and, thus, is in $\mathcal{B}_i$. By evaluating $\tilde{q}_{i,1}(x)$ at all elements of $\bar{\mathcal{R}}$, of which there are $(n-k)$, we determine the number of roots of this type, which we denote by $p_i$. We note here that, since there is no restriction on the placement of roots in the set $\mathcal{B}_i$ (i.e., there may be roots shared by $\tilde{q}_{i,0}(x)$ and $\tilde{q}_{i,1}(x)$ which fall in $\mathcal{R}$) we have not, necessarily, determined the entire set. However, we make the assumption that we have completely determined $\mathcal{B}_i$ and, thus, any remaining roots of $\tilde{q}_{i,1}(x)$ must be in the set $\mathcal{A}_i$. Our assumption leads us to an estimate of the cardinality of $\mathcal{A}_i$ as,

$$|\hat{\mathcal{A}}_i| = (\deg(\tilde{q}_{i,1}(x)) - p_i). \qquad (62)$$

Our interest in the placement of the roots of $\tilde{q}_{i,1}(x)$ is based on our selection of the indices in $\mathcal{J}$. By *Theorem 1*, the *evaluation-map* encoding of $\hat{m}_i(x)$ must have Hamming distance no greater than $(n-k+1)/2$ from the hard-decision vector. However, because the first step in the interpolation procedure is to modify all test-vectors, making them zero on all coordinate positions in $\mathcal{J}$, the hard-decision vector specified by *Theorem 1* is $\mathbf{y}_i'$. In (60), the multiplicative factor $v(x)$ acts to ensure that the *evaluation-map* encoding of $\hat{m}_i(x)$ is zero on the coordinate locations in $\mathcal{J}$, a fact which is made evident using the definition of $v(x)$ (28)

$$\hat{m}_i(x) = \frac{\tilde{q}_{i,0}(x)}{\tilde{q}_{i,1}(x)} \cdot \prod_{j \in \mathcal{J}} (x + \alpha_j). \qquad (63)$$

The *evaluation-map* encoding of $\hat{m}_i(x)$ will, therefore, be zero on all indices in $\mathcal{J}$, provided no factors of $v(x)$ are canceled by factors of $\tilde{q}_{i,1}(x)$ (i.e., $\mathcal{A}_i$ is the empty set). If $\mathcal{A}_i$ is not empty, then each $\alpha_m \in \mathcal{A}_i$ cancels the corresponding factor $(x+\alpha_m)$ from $v(x)$, thus indicating that a hard-decision error has occurred in coordinate position $m$. (Although we recognize that, if $\alpha_m \in \mathcal{B}_i$, then this factor is effectively replaced, we have observed that the intersection of $\mathcal{A}_i$ and $\mathcal{B}_i$ is almost always empty and, thus, we do not consider this case.) Because we have chosen the coordinate positions of $\mathcal{J}$ as being those of highest reliability, the probability that $\hat{m}_i(x)$ is the true message polynomial is inversely related to $|\mathcal{A}_i|$, which may be accurately estimated using (62).

```
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) $\Gamma = \{i;\ d_{0,i} < d_{1,i},\ d_{0,i} = d_0^*\}$
  3) $i^* = \arg\max_{i \in \Gamma}(w_i)$
  4) $Q^*(x,z) = Q_{i^*}(x,z)$
OUTPUT
  $Q^*(x,z)$
```

The procedure we use to determine the single element of $\mathcal{Q}$, denoted as $Q^*(x,z)$, which is factored into the decoded message polynomial is given as **Algorithm 3**. In such, we first evaluate each denominator polynomial $\tilde{q}_{i,1}(x)$ at the $(n-k)$ field elements in $\bar{\mathcal{R}}$ to determine the number of roots $p_i$ contained in this set. Since these roots must be common to $\tilde{q}_{i,0}(x)$ and $\tilde{q}_{i,1}(x)$, we determine the remaining degrees after dividing out these common factors as

$$d_{0,i} = \deg(\tilde{q}_{i,0}(x)) - p_i \qquad (64)$$
$$d_{1,i} = \deg(\tilde{q}_{i,1}(x)) - p_i. \qquad (65)$$

Using the following inequality,

$$\deg(\tilde{q}_{i,1}(x)) > \deg(\tilde{q}_{i,0}(x)), \qquad (66)$$

which is clear from $\deg(v(x)) = k$ and $\deg(\hat{m}_i(x)) < k$, we note that $d_{0,i}$ is upper bounded by $d_{1,i}$, our estimate of the number of hard-decision errors occurring in the indices in $\mathcal{J}$. We, thus, choose $Q^*(x,z)$ as the element of $\mathcal{Q}$ that minimizes this value. In the event that multiple polynomials have equal, minimal values of $d_{0,i}$, we choose the polynomial associated with the test-vector of highest *a-posteriori probability*. Since all test-vectors are equivalent on the indices in $\bar{\mathcal{I}}$, this computation need only be conducted among the $\eta$ indices in $\mathcal{I}$.

## IV. Computational Complexity

In this section, we provide computational complexity results in terms of the maximum number of multiplications required to decode a single frame. This figure of merit was chosen because a typical hardware implementation of decoding over an extension of the binary field represents field elements as degree-$(q-1)$ polynomials over GF(2). As such, addition is easily implemented using $q$ exclusive-or operations. However, multiplication requires a convolution of polynomials followed by a reduction modulo an irreducible polynomial of degree-$q$.

The majority of required computation is, thus, used executing multiplications.

Although a full analysis of the computational complexity of the presented algorithm is beyond the scope of this paper (a complete treatment is provided in [19]), our intention is to provide results such that a comparison to existing decoding methodologies can be made. In particular, we compare our technique with HDD, in which *The Berlekamp-Massey Algorithm* [5] is utilized to solve the *Key-Equations*, and *Kötter-Vardy Decoding* [7], [15], with parameters chosen to obtain a comparable performance to the presented algorithm. We provide the complexity results as a function of the code parameters and $\eta$. From this discussion, we also provide a practical range of values from which $\eta$ is selected to obtain a sufficient coding gain over hard-decision decoding without a significant increase in decoding complexity.

Following the complexity analysis in [19], the number of multiplications required to perform test-set modification (see Section III-A), which is characterized by the required erasures-only decoding performed on the MAP hard-decision vector, is bounded as,

$$Mult_{\text{Test-Set Modification}} \leq 3(n-k)^2. \tag{67}$$

The complexity of *Common-Element Interpolation* (**Algorithm 1**) and *Uncommon-Element Interpolation* (**Algorithm 2**) are respectively bounded as

$$\begin{aligned} Mult_{\text{Common-Element}} &\leq 3(n-k-\eta)^2 + \\ &\quad (n-k-\eta)\cdot(\eta+3). \quad (68)\\ Mult_{\text{Uncommon-Element}} &\leq (2^\eta - 1)\cdot(5(n-k)-3) + \\ &\quad \eta\cdot(n-k+6). \quad (69) \end{aligned}$$

We, finally, bound the complexity of RCF, which consists both of polynomial selection (**Algorithm 3**) and evaluation of the decoded message polynomial using (60), as

$$Mult_{\text{RCF}} \leq (n-k)\cdot(2^{(\eta-1)}\cdot(n-k)+k), \tag{70}$$

producing the following total LCC complexity bound;

$$\begin{aligned} Mult_{\text{LCC}} \leq\ & (n-k)^2\cdot(6+2^{(\eta-1)}) + \\ & (n-k)\cdot(5\cdot 2^\eta + k - 4\eta - 2) - \\ & 3\cdot 2^\eta + 2\eta^2 + 3\eta + 3. \quad (71) \end{aligned}$$

In order to make a comparison of the complexity of LCC decoding to HDD, in which the *Berlekamp-Massey Algorithm* is employed to solve the *Key-Equations*, we must also establish a bound on its complexity. As given in [19], the complexity of traditional HDD is easily evaluated as the sum of its constituent parts; *1) Syndrome Polynomial Computation*, *2) Error-Locator Polynomial Computation using the B-M Algorithm*, *3) Error-Evaluator Polynomial Computation*, *4) Chien-Search*, *5) Error-Locator Polynomial Derivative Computation*, and *6) Forney's Algorithm* (a full discussion of each step of the HDD algorithm and the associated complexity may be found in [2]). By summing the maximum number of multiplications required for each step of HDD, we obtain the following bound

$$Mult_{\text{HDD-BM}} \leq 7(n-k)^2 + (n-k)\cdot(3n+1)/2. \tag{72}$$

TABLE I
UPPER BOUND ON THE NUMBER OF MULTIPLICATIONS REQUIRED FOR TO DECODE A SINGLE FRAME FOR LCC, HDD W/ BERLEKAMP-MASSEY, GMD, AND *K-V* DECODING*.

| ALGORITHM | $\mathcal{C}_5(31,25)$ | $\mathcal{C}_6(63,55)$ | $\mathcal{C}_8(255,239)$ | $\mathcal{C}_{10}(460,420)$ |
|---|---|---|---|---|
| HDD w/ BM | **534** | **1208** | **7920** | **38820** |
| GMD** Decoding | **546** | **1215** | **7803** | – |
| *K-V* (mm = 4) | - | **36000** | **250000** | - |
| LCC ($\eta = 1$) | **428** | 922 | 5682 | 28162 |
| LCC ($\eta = 2$) | **503** | **1037** | 6037 | 30005 |
| LCC ($\eta = 3$) | **672** | **1294** | **6806** | 33846 |
| LCC ($\eta = 4$) | 1025 | **1831** | **8399** | 41679 |
| LCC ($\eta = 5$) | 1742 | 2924 | **11636** | **57492** |
| LCC ($\eta = 6$) | 3183 | 5125 | 18157 | **89262** |

* Values shown in boldface type correspond to simulation results given in Section V.
** Values provided reflect the complexity of the GMD approximation presented in [11].

Much recent work has been related to reducing the complexity of GMD decoding of RS and BCH codes. A significant portion of this work, however, has focused on low-complexity methods of producing the numerous *errata-locator* polynomials required for such decoding (see [10]). Since a significant portion of the GMD complexity lies in evaluating the roots of these polynomials, the overall complexity of these methods remains (comparatively) high. An approximation to GMD decoding, however, was presented in [11], which assuages this problem at the cost of a reduction in decoding performance over traditional GMD decoding. The complexity results provided in Table I reflect this approximate GMD decoding methodology.

In Table I, we provide a complexity comparison of LCC (with RCF), HDD utilizing the *B-M Algorithm*, GMD decoding, and *K-V* decoding for the following codes; $\mathcal{C}_5(31,25)$, $\mathcal{C}_6(63,55)$, $\mathcal{C}_8(255,239)$, $\mathcal{C}_{10}(460,420)$. We give results for $\eta = 1, 2, \ldots, 6$, however, only the values given in boldface type are considered in the simulation results provided in the following section. We consider these values of $\eta$ to provide a significant performance gain over classical HDD without a large ($< 2.5\times$) increase in complexity. As can be seen, choosing $\eta \approx \log_2(n-k)+1$ allows for this performance vs. complexity tradeoff to be attained.

The maximum multiplicity used for the presented *K-V* decoding procedures were chosen to allow a comparable performance to the considered LCC algorithm (see Section V). Here, we note that the complexity of *K-V* decoding is highly dependent on the method with which it is implemented. The required multiplications for *K-V* decoding given in Table I are based on the *Reduced-Complexity Interpolation* and *Reduced-Complexity Factorization* discussions presented in [15] and, thus, are the lowest values that we are aware of.

## V. SIMULATIONS RESULTS

The frame error-rates (FERs) for LCC decoding and classical HDD are given in Figures 2-5 for $\mathcal{C}_5(31,25)$, $\mathcal{C}_6(63,55)$, $\mathcal{C}_8(255,239)$, and $\mathcal{C}_{10}(460,420)$, respectively. Here, results are provided for both full-factorization, as would be obtained with traditional Chase decoding, and for *Reduced-Complexity Factorization* (RCF). We also provide simulation results for *K-V* decoding (Figures 3-4), with a maximum multiplicity of 4, for the $\mathcal{C}_6(63,55)$ and $\mathcal{C}_8(255,239)$ codes, and GMD decoding

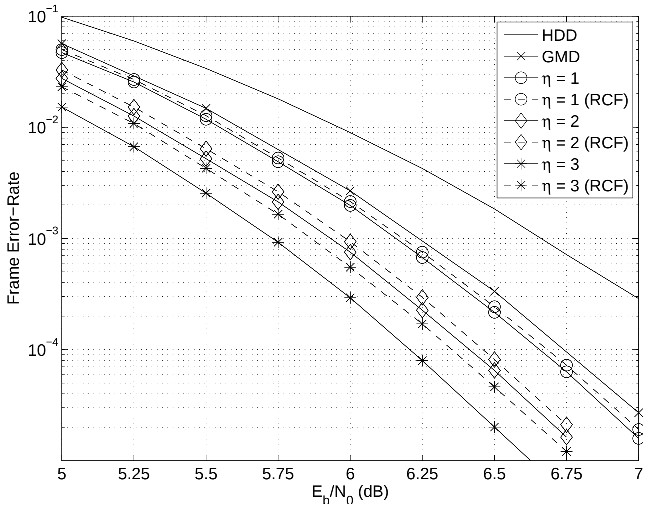


Fig. 2. Frame-Error Rate of $\mathcal{C}_5(31,25)$ for HDD, GMD, and LCC decoding, $\eta = 1, 2, 3$, as used over an AWGN channel with BPSK modulation.

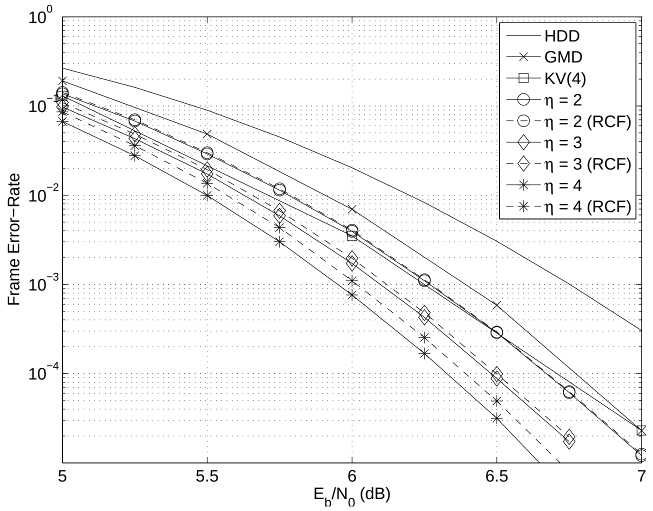


Fig. 3. Frame-Error Rate of $\mathcal{C}_6(63,55)$ for HDD, GMD, *K-V* (maximum multiplicity = 4), and LCC decoding, $\eta = 2, 3, 4$, as used over an AWGN channel with BPSK modulation.

for the $\mathcal{C}_5(31,25)$, $\mathcal{C}_6(63,55)$, and $\mathcal{C}_8(255,239)$ (Figures 2-4) codes. All FER plots are given as a function of $E_b/N_0$ (see (4)), with the exception of $\mathcal{C}_{10}(460,420)$. Because this code is widely utilized in the magnetic recording industry, we have presented these results as a function of the inverse of the noise variance (assuming unit signal energy), a traditional figure-of-merit for such applications. All results provided are assuming the use of binary phase-shift keying modulation over an additive white Gaussian noise channel.

We first note that the gains achieved by GMD RS decoding are between .5 and .75 times the gains achieved LCC decoding for the smallest values of $\eta$ considered (see Figures 2-4). The gains achieved by the GMD approximation presented in [11] are further reduced to about 3/4 those obtained by full GMD decoding. As shown in Table I, the complexity of this GMD approximation are 15-30% higher than that of LCC decoding for the smallest values of $\eta$ considered. It is, thus, clearly advantageous to utilize LCC decoding over either GMD RS decoding or its approximation.

We, secondly, note that significant coding gains over HDD are attainable using both LCC and *K-V* decoding (see Figures 3-4). The performance achieved by *K-V* decoding, which is similar to the presented LCC decoding performance, however, is obtained at a significantly higher complexity than required for either HDD or LCC decoding (see Table I). Since comparable decoding performance is, clearly, achievable using LCC decoding, at a fraction of the complexity, we deem it sufficient to only compare HDD with LCC in the following.

As can be seen, LCC decoding offers an increase in decoding performance of over HDD of .8 dB, .75 dB, .5 dB, and .3 dB for $\mathcal{C}_5(31,25)$, $\mathcal{C}_6(63,55)$, $\mathcal{C}_8(255,239)$, and $\mathcal{C}_{10}(460,420)$, respectively. We obtain these performance gains with less than 2.5 times the computation required for HDD. In many cases, significant gains are achievable at a lower decoding complexity than HDD. Although these performance gains are, clearly, decreasing with code length, we expect this behavior from any Chase-type algorithm. Chase decoding is, essentially, a pseudo-extension of the decoding radius of HDD in that as many as $(d_{\min}/2+\eta)$ symbol errors may be tolerated by the algorithm, however, no guarantee is made that more than $d_{\min}/2$ errors will be tolerated. Therefore, because a longer code of an equivalent rate will have a larger decoding radius, larger values of $\eta$ are required to significantly improve the performance of longer codes. Since the decoding complexity of LCC is exponential in $\eta$, achieving significant gains at a practical complexity is made more difficult with increasing code length. We do note, however, that the gain offered by LCC is still significant for code lengths as large 460 symbols over GF(1024) (4600 bits), in which LCC decoding is less than 2.5 times as complex as HDD.

We next discuss the suboptimality of *Reduced-Complexity Factorization* (RCF), as compared to full-factorization. This suboptimality, which is most evident in Figures 2 and 3, is clearly increasing with $\eta$. This trend is explained by the fact that RCF attempts to select a single polynomial which will decode correctly from the set of interpolation polynomials $\mathcal{Q}$. As $\eta$ is increased, the size of $\mathcal{Q}$ (which is $2^\eta$) grows exponentially, making the selection process more difficult. However, simulations show that the performance loss incurred by RCF is negligible for long codes ($n \geq 255$), such that no perceivable loss is incurred by $\mathcal{C}_8(255,239)$ and $\mathcal{C}_{10}(460,420)$ for values of $\eta$ up to 6. In fact, the event in which RCF and full-factorization produce different decoded messages was never observed during the simulation of the $\mathcal{C}_{10}(460,420)$ code. Thus, provided our code length is large enough, RCF will not incur a performance degradation, even for values of $\eta$ exceeding the practical limit of $\eta \approx \log_2(n-k)+1$ (Section IV).

## VI. Conclusions

In this work we have presented an algebraic procedure that implements low-complexity Chase-type decoding of RS

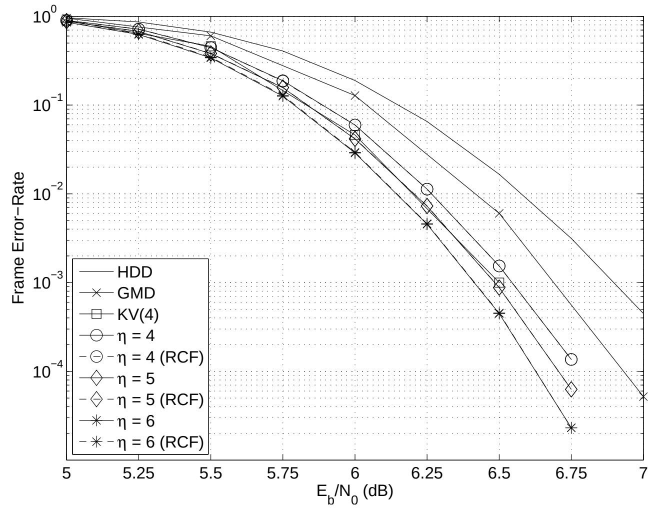


Fig. 4. Frame-Error Rate of $\mathcal{C}_8(255, 239)$ for HDD, GMD, *K-V* (maximum multiplicity = 4), and LCC decoding, $\eta = 4, 5, 6$, as used over an AWGN channel with BPSK modulation.

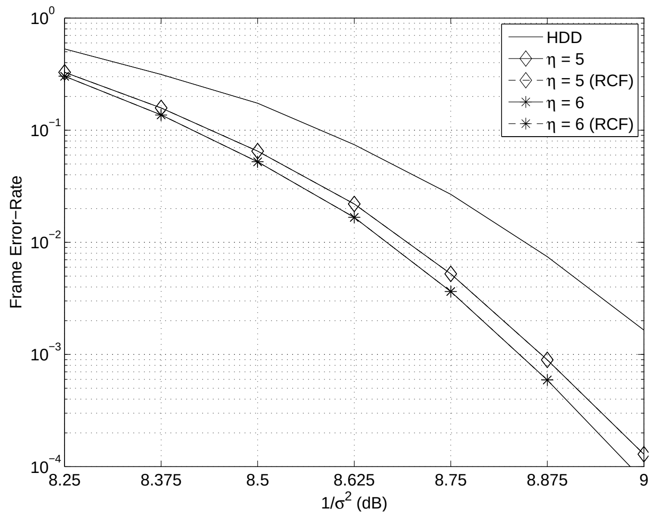


Fig. 5. Frame-Error Rate of $\mathcal{C}_{10}(460, 420)$ for HDD and LCC decoding, $\eta = 5, 6$, as used over an AWGN channel with BPSK modulation.

codes. The complexity reduction achieved by this algorithm is obtained by an interpolation technique that exploits the similarity among test-vectors, and by a reduced-complexity approximation to the full-factorization of each interpolation polynomial. The performance-vs.-complexity tradeoffs of the low-complexity Chase (LLC) decoder can be summarized as follows.

1) By choosing the Chase parameter $\eta = 0$ (i.e. the test-set consists only of the MAP hard-decision vector), we achieve the performance of hard-decision decoding (HDD) (which is equivalent to Sudan decoding with multiplicity = 1, as given by *Theorem 1*), but with a slightly reduced complexity than that required for the traditional Berlekamp-Massey HDD.
2) By choosing the Chase parameter $\eta$ appropriately, we can get the complexity similar to the Berlekamp-Massey HDD, but achieve a significant coding gain over HDD.
3) By choosing the Chase parameter $\eta$ to be such that the LLC decoding complexity is within 2.5x the complexity of of the Berlekamp-Massey HDD, we achieve performance equal (or even better) than the Kötter-Vardy algorithm with multiplicity 4 - KV(4). Thereby, the decoding complexity of the LLC decoder is still a fraction of the decoding complexity required for KV(4).

## References

[1] G. Reed and I. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, pp. 300-304, 1960.
[2] S. Wicker and V. Bhargava, *Reed-Solomon Codes and Their Applications*. IEEE Press, 1994.
[3] W. Peterson, "Encoding and error-correction procedures for Bose-Chaudhuri codes," *IRE Transactions on Information Theory*, vol. IT-60, pp. 459-470, 1960.
[4] E.R. Berlekamp, "Nonbinary BCH Decoding," *IEEE Transactions on Information Theory*, vol. IT-14, p. 242, 1968.
[5] J.L. Massey, "Shift register synthesis and BCH decoding," *IEEE Transactions on Information Theory*, vol. IT-15, pp. 122-127, January 1968.
[6] M. Sudan, "Decoding of Reed Solomon codes beyond the error-correction bound," *Journal of Complexity*, vol. 13, no. 1, pp. 180-193, 1997.
[7] R. Kötter and A. Vardy, "Algebraic soft-decision decoding of Reed-Solomon codes", 2000.
[8] D. Forney Jr., "Generalized minimum distance decoding," *IEEE Transactions on Information Theory*, vol. IT-12, no. 2, pp. 125-131, 1966.
[9] D. Chase, "A class of algorithms for decoding block codes with channel measurement information," *IEEE Transactions on Information Theory*, vol. IT-18, pp. 170-182, January 1972.
[10] D. Taipale and M. Seo, "An efficient soft-decision Reed-Solomon decoding algorithm," *IEEE Transactions on Information Theory*, vol. IT-40, no. 4, pp. 1130-1139, 1994.
[11] N. Kamiya, "On acceptance criterion for efficient successive errors-and-erasures decoding of Reed-Solomon and BCH codes", *IEEE Transactions on Information Theory*, vol. IT-43, no. 5, pp. 1477-1488, 1997.
[12] N. Kamiya, "On algebraic soft-decision decoding algorithms for BCH codes", *IEEE Transactions on Information Theory*, vol. IT-47, no. 1, pp. 45-58, 2001.
[13] H. Xia, C. Zhong, and J. Cruz, "A Chase-type algorithm for soft-decision Reed-Solomon decoding on Rayleigh fading channels", *Proceedings of the IEEE Global Communications Conference (GlobeCom)*, pp. 1751-1755, 2003.
[14] H. Xia and J. Cruz, "Reliability-based forward recursive algorithms for algebraic soft-decision decoding of Reed-Solomon codes", *IEEE Transactions on Communications*, vol. IT-55, no. 7, pp. 1273-1278, 1997.
[15] W. J. Gross, F. R. Kschischang, R. Kötter, and P.G. Gulak, "Towards a VLSI architecture for interpolation-based soft-decision Reed-Solomon decoders", *Journal of VLSI Signal processing*, July 2003.
[16] R. E. Blahut, *Algebraic Codes for Data Transmission*. Cambridge University Press, 2003.
[17] S. Haykin, *Communication Systems: 4th Edition*. Wiley and Sons, 2001.
[18] R. Nielsen, *Decoding AG-codes beyond half the minimum distance*. PhD Thesis, Technical University of Denmark, 1998.
[19] J. Bellorado, *Low-complexity soft decoding Algorithms for Reed-Solomon Codes*. PhD Thesis, Harvard University, 2006.
[20] R. Roth, and G. Ruckenstein, "Efficient decoding of Reed-Solomon codes beyond half the minimum distance", *IEEE Transactions on Information Theory*, vol. 46, pp. 246-257, January 2000.

**Jason Bellorado** received the degree of Bachelor of Science in Electrical Engineering from University of Massachusetts, Lowell, in 2000, where he was awarded the Dean's Medal for the highest academic standing in the James B. Francis College of Engineering. Under the advisement of Professor Aleksandar Kavčić, he received the degrees of Master of Arts (2001) and

Ph.D. (2006) in Engineering Sciences from the *Division of Engineering and Applied Sciences* (DEAS) at Harvard University in Cambridge, Massachusetts.

In March, 2006, Dr. Bellorado joined Link-A-Media Devices (LAMD) Corporation, a semiconductor startup company in Santa Clara, California, which specializes in information storage. He has worked as a systems architect for both hard-disk drives and solid-state drives, and currently holds patents in advanced digital signal processing and signal detection methods, robust signal timing in low-SNR conditions, error control coding techniques, and read-channel optimization procedures which span both fields. In the 4 years that Jason has been working for LAMD he has been promoted to the rank of *Senior Manager* and currently manages a team of both read-channel and validation engineers.

**Aleksandar Kavčić** received the Dipl. Ing. degree in Electrical Engineering from Ruhr-University, Bochum, Germany in 1993, and the Ph.D. degree in Electrical and Computer Engineering from Carnegie Mellon University, Pittsburgh, Pennsylvania in 1998.

Since 2007 he has been with the University of Hawaii, Honolulu where he is presently Associate Professor of Electrical Engineering. Prior to 2007, he was in the Division of Engineering and Applied Sciences at Harvard University, as Assistant Professor of Electrical Engineering from 1998 to 2002, and as John L. Loeb Associate Professor of Natural Sciences from 2002 to 2006. While on leave from Harvard University, he served as Visiting Associate Professor at the City University of Hong Kong in the Fall of 2005 and as Visiting Scholar at the Chinese University of Hong Kong in the Spring of 2006.

Prof. Kavčić received the IBM Partnership Award in 1999 and the NSF CAREER Award in 2000. He is a co-recipient, with X. Ma and N. Varnica, of the 2005 IEEE Best Paper Award in Signal Processing and Coding for Data Storage. He served on the Editorial Board of the IEEE Transactions on Information Theory as Associate Editor for Detection and Estimation from 2001 to 2004, as Guest Editor of the IEEE Signal Processing Magazine in 2003-2004, and as Guest Editor of the IEEE Journal on Selected Areas in Communications in 2008-2009. From 2005 until 2007, he was the Chair of the Data Storage Technical Committee of the IEEE Communications Society.