Algebraic Soft-Decision Decoding of Reed-Solomon Codes

Ralf Koetter
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL 61801, U.S.A.
`koetter@shannon.csl.uiuc.edu`

Alexander Vardy
Department of Electrical Engineering
University of California, San Diego
La Jolla, CA 92093, U.S.A.
`vardy@kilimanjaro.ucsd.edu`

August 31, 2001

Abstract

A polynomial-time soft-decision decoding algorithm for Reed-Solomon codes is developed. This list-decoding algorithm is algebraic in nature and builds upon the interpolation procedure proposed by Guruswami-Sudan for hard-decision decoding. Algebraic soft-decision decoding is achieved by means of converting the probabilistic reliability information into a set of interpolation points, along with their multiplicities. The proposed conversion procedure is shown to be asymptotically optimal for a certain probabilistic model. The resulting soft-decoding algorithm significantly outperforms both the Guruswami-Sudan decoding and the generalized minimum distance (GMD) decoding of Reed-Solomon codes, while maintaining a complexity that is polynomial in the length of the code. Asymptotic analysis for a large number of interpolation points is presented, leading to a geometric characterization of the decoding regions of the proposed algorithm. It is then shown that the asymptotic performance can be approached as closely as desired with a list-size that does not depend on the length of the code.

Keywords: Berlekamp-Welch algorithm, list decoding, polynomial interpolation, Reed-Solomon codes, soft-decision decoding, Guruswami-Sudan algorithm

Work performed while both authors were on leave at Laboratoire I3S, C.N.R.S., Sophia-Antipolis, France. Research supported in part by the David and Lucile Packard Foundation.

1. Introduction

Reed-Solomon (RS) codes are among the most extensively used error-control codes, with applications ranging from magnetic recording [13], through satellite communications [4, 29], to deep-space exploration [16]. An important problem in hard-decision decoding of Reed-Solomon codes is that of decoding beyond the error-correction radius. A breakthrough in this area was achieved by Sudan [25, 11]. In the form presented in [11], the algorithm of Guruswami-Sudan corrects any fraction of $\tau \leqslant 1 - \sqrt{R}$ erroneous positions in a Reed-Solomon code of rate R. Thus the error-correction capability of the algorithm exceeds the minimum distance bound $(1-R)/2$ for all rates in the interval $[0,1)$.

Soft-decision decoding of Reed-Solomon codes is, however, an entirely different matter. Even though the decoder can be often supplied with reliable soft-decision data relatively easily [4], the high complexity of optimal soft-decision decoding makes full utilization of such data prohibitive. Indeed, all the available *optimal* soft-decoding algorithms for RS codes, such as [27] and its modifications [6, 21, 22], are non-algebraic and run in time that scales *exponentially* with the length of the code. This makes the use of such algorithms generally infeasible in practice. An alternative approach to the problem of efficient soft decoding, pioneered by Forney [9, 10], is known as generalized minimum distance (GMD) decoding. While the complexity of GMD decoding is moderate, and ultimately is of the same order as the complexity of hard-decision decoding [2, 14, 15, 24], the gains that can be realized by GMD decoding are also moderate (cf. Figure 1). Thus, in light of the ubiquity of Reed-Solomon codes, efficient soft-decision decoding of RS codes is one of the most important problems in coding theory and practice.

Our goal in this paper is to present an efficient soft-decision decoding algorithm for Reed-Solomon codes. The algorithm significantly outperforms both the Guruswami-Sudan list decoding [11] and the GMD-based decoding methods. For example, Figure 1 shows the performance of the three algorithms for a simple coding scheme: codewords of the $(255, 144, 112)$ Reed-Solomon code over GF(256) are modulated using a 256-QAM signal constellation and transmitted over an AWGN channel. More details on this can be found in the caption of Figure 1 and in Section 6. We note that similar coding schemes, although with higher-rate RS codes, are in use today on satellite communication channels.

The proposed algorithm is based on the algebraic interpolation techniques developed by Sudan [11, 25]. To achieve soft-decision decoding, we translate the soft-decision reliability information provided by the channel into a set of algebraic constraints. Specifically, given the channel output vector $(y_1, y_2, \ldots, y_n)$ and the a posteriori transition probabilities $\Pr(c_i|y_i)$, we iteratively compute a set of interpolation points along with their multiplicities. We show that, at each step of the computation, this choice of interpolation points is optimal, in a certain precise sense defined in Section 4.

Notably, the algebraic interpolation and factorization techniques of Guruswami-Sudan [25, 11] can be implemented efficiently in polynomial time. There has been a lot of research on this topic recently [1, 7, 8, 19, 18, 23, 30]. Our soft-decision decoding procedure inherits these properties of Guruswami-Sudan decoding. In addition, one of the most useful characteristics of our soft-decoding algorithm is a complexity/performance trade-off that

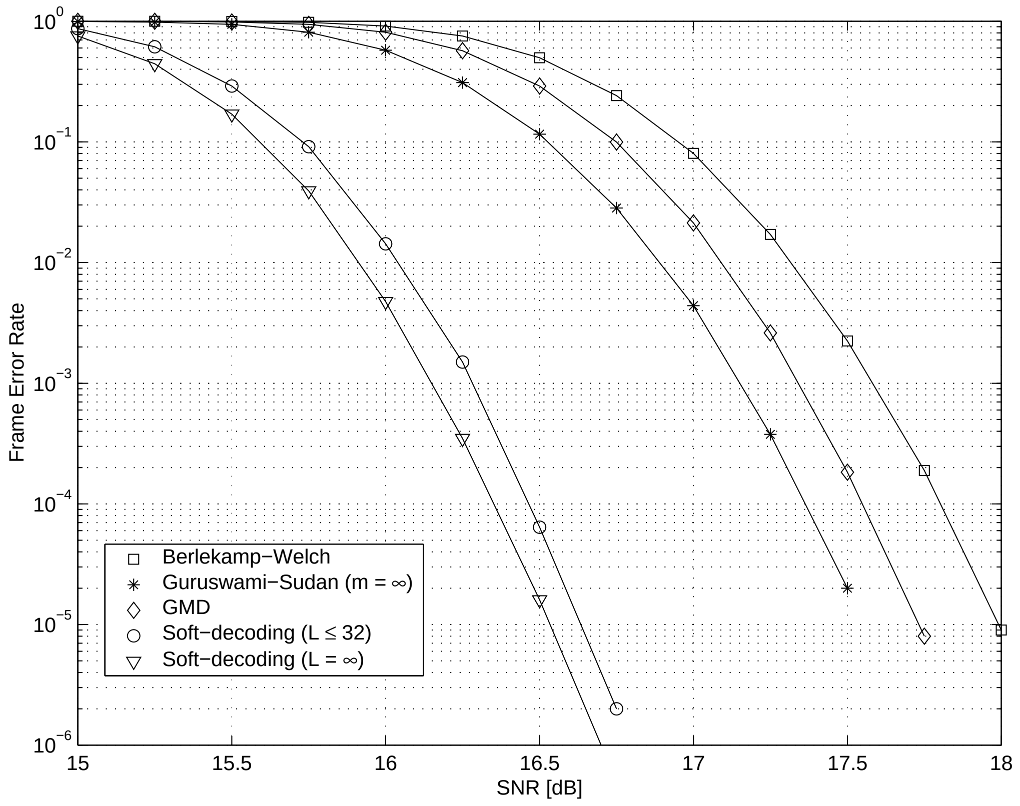

Figure 1. *Performance comparison for a simple coding scheme*

Codewords of the $(255,144,112)$ Reed-Solomon code are modulated using a 256-QAM signal constellation and transmitted over an additive white Gaussian noise (AWGN) channel. At the channel output, soft decisions are quantized to 8 bits. The different curves correspond to the performance achieved by two hard-decision decoding algorithms and two soft-decision decoding algorithms. The two hard-decision algorithms are the conventional Berlekamp-Welch [28] decoding up to half the minimum distance and the list-decoding algorithm by Guruswami-Sudan [11]. For the latter, asymptotic performance is shown, assuming that the multiplicity of interpolation points tends to infinity (cf. Theorem 2). The two soft-decision algorithms are Forney's GMD decoding [9] and the algebraic soft-decision list-decoding algorithm developed herein. The curve marked $\triangledown$ describes asymptotic performance for a large number of interpolation points, and hence large list-size. However, the curve marked $\circ$ shows that the asymptotic performance can be closely approached with a finite list that is guaranteed to have at most 32 codewords (cf. Section 6).

can be chosen freely. In particular, the complexity can be adjusted to any required level of performance within a certain fundamental bound (cf. Theorem 12).

The rest of this paper is organized as follows. The next section contains a brief overview of Sudan's list-decoding algorithm, as presented in [11]. Section 3 then sets the ground for algebraic soft-decision decoding of Reed-Solomon codes. In particular, we define the concepts of *score* and *cost* associated with each possible set of interpolation points. We then give a sufficient condition for successful list-decoding in terms of these concepts. The core of our soft-decoding algorithm is developed in Section 4, which deals with the computation of (the multiplicities of) the interpolation points. In particular, we show how to iteratively compute the interpolation *multiplicity matrix* so as to maximize the expected score in a certain probabilistic model. We prove that the greedy approach produces such a matrix at each step of the computation. Section 5 presents an asymptotic performance analysis for our algorithm as the the number of interpolation points approaches infinity. The analysis leads to a simple geometric characterization of the (asymptotic) decoding regions of our algorithm. In Section 6, we show that the asymptotic performance can be approached arbitrarily closely with a list size that depends on the rate but *not* on the the length of the code at hand. We also present simulation results for various list sizes, for both half-rate and high-rate Reed-Solomon codes. Finally, we conclude with a brief discussion in Section 7.

2. Reed-Solomon codes and the Sudan algorithm

We first set up some of the notation that will be used throughout this work. Let $\mathbb{F}_q$ be the finite field with q elements. The ring of polynomials over $\mathbb{F}_q$ in a variable X is denoted $\mathbb{F}_q[X]$. Reed-Solomon codes are obtained by evaluating certain subspaces of $\mathbb{F}_q[X]$ in a set of points $\mathcal{D} = \{x_1, x_2, \ldots, x_n\}$ which is a subset of $\mathbb{F}_q$. Specifically, the Reed-Solomon code $\mathbb{C}_q(n,k)$ of length n and dimension k is defined as follows:

$$\mathbb{C}_q(n,k) \overset{\text{def}}{=} \{\, (f(x_1),\ldots,f(x_n)) \;:\; x_1,\ldots,x_n \in \mathcal{D},\; f(X) \in \mathbb{F}_q[X],\; \deg f(X) < k \,\} \quad (1)$$

The point set $\mathcal{D}$ is usually taken as $\mathbb{F}_q$ or as $\mathbb{F}_q^*$, the set of all the nonzero elements of $\mathbb{F}_q$. The set of polynomials of degree less than k in $\mathbb{F}_q[X]$ is a linear space, which together with the linearity of the evaluation map (1) establishes that $\mathbb{C}_q(n,k)$ is a linear code. The minimum Hamming distance of $\mathbb{C}_q(n,k)$ is $d = n-k+1$, which follows from the fact that any nonzero polynomial of degree less than k evaluates to zero in less than k positions.

Given an arbitrary vector $\underline{y} \in \mathbb{F}_q^n$, the ***hard-decision decoding task*** consists of finding the codeword $\underline{c} \in \mathbb{C}_q(n,k)$ such that the Hamming weight $\mathrm{wt}(\underline{e})$ of the error vector $\underline{e} = \underline{y} - \underline{c}$ is minimized. The Berlekamp-Welch algorithm [28] is a well-known algorithm that accomplishes this task, provided $\mathrm{wt}(\underline{e}) < d/2$. Generalizing upon Berlekamp-Welch [28], Sudan [25] and Guruswami-Sudan [11] derived a polynomial-time algorithm that achieves error correction substantially beyond half the minimum distance of the code. In the remainder of this section, we describe the essential elements of this algorithm.

Definition 1. *Let $\mathcal{A}(X,Y) = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} a_{i,j}X^iY^j$ be a bivariate polynomial over $\mathbb{F}_q$ and let w_X, w_Y be nonnegative real numbers. The (w_X,w_Y)-**weighted degree** of $\mathcal{A}(X,Y)$, denoted $\deg_{w_X,w_Y}\mathcal{A}(X,Y)$, is the maximum over all numbers $iw_X + jw_Y$ such that $a_{i,j} \neq 0$.*

The $(1,1)$-weighted degree is simply the ***degree*** of a bivariate polynomial. The number of monomials of (w_X,w_Y)-weighted degree at most δ is denoted $N_{w_X,w_Y}(\delta)$. Thus

$$N_{w_X,w_Y}(\delta) \overset{\text{def}}{=} \big|\big\{X^iY^j \;:\; i,j \geqslant 0 \text{ and } iw_X + jw_Y \leqslant \delta\big\}\big|$$

The following lemma provides a closed-form expression for $N_{w_X,w_Y}(\delta)$ for the case $w_X = 1$. Similar statements can be found in [11, 18, 23, 25] and other papers.

Lemma 1.
$$N_{1,k}(\delta) \;=\; \left\lceil \frac{\delta+1}{k} \right\rceil \left(\delta - \frac{k}{2} \left\lfloor \frac{\delta}{k} \right\rfloor + 1 \right)$$

The lemma follows by a straightforward counting of monomials; for a proof, see [11, Lemma 6]. The exact expression in Lemma 1 can be converted into a simple lower bound:

$$N_{1,k-1}(\delta) \;=\; \frac{(\delta+1)^2}{2(k-1)} + \frac{k-1}{2}\left(\left\lceil \frac{\delta+1}{k-1} \right\rceil - \left(\left\lceil \frac{\delta+1}{k-1} \right\rceil - \frac{\delta+1}{k-1} \right)^2 \right) \;>\; \frac{\delta^2}{2(k-1)} \qquad (2)$$

This is, in fact, a special case of the more general lower bound $N_{w_X,w_Y}(\delta) > \delta^2/2w_X w_Y$. The latter bound can be easily proved using geometric arguments, as shown in Figure 2.

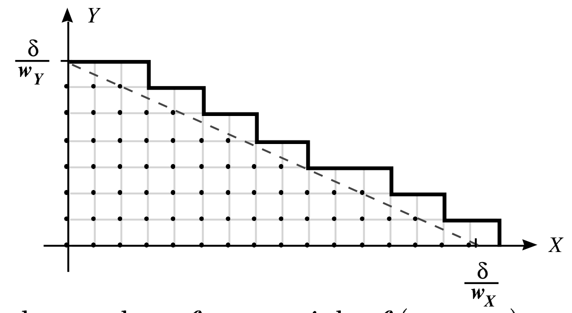

Figure 2. *A bound on the number of monomials of* (w_X, w_Y)*-weighted degree at most* δ

Here $N_{w_X,w_Y}(\delta)$ is the area under the solid line, with each monomial $X^a Y^b$ represented by the unit square whose lower left corner is at the point (a,b). It is easy to see that the triangle of area $\delta^2/2w_X w_Y$ (bounded by the dashed line) is completely enclosed by the solid line.

Given the channel output vector $\underline{y} = \underline{c} + \underline{e} = (y_1, y_2, \ldots, y_n)$ and the corresponding point set $\mathcal{D} = \{x_1, x_2, \ldots, x_n\}$, we consider the set of pairs $\mathcal{P} = \{(x_1,y_2),(x_2,y_2),\ldots,(x_n,y_n)\}$ as ***points*** in a two-dimensional affine space. Given a point (α,β) and a bivariate polynomial $\mathcal{A}(X,Y)$, we say that (α,β) ***lies on*** $\mathcal{A}(X,Y)$ if $\mathcal{A}(\alpha,\beta) = 0$. Equivalently, we say that $\mathcal{A}(X,Y)$ ***passes through*** the point (α,β). Herein, we will be interested in bivariate polynomials that not only pass through all the points in $\mathcal{P}$ but do so with high multiplicities.

Definition 2. *A bivariate polynomial $\mathcal{A}(X,Y)$ is said to pass through a point (α,β) with multiplicity m if the shifted polynomial $\mathcal{A}(X+\alpha, Y+\beta)$ contains a monomial of degree m and does not contain a monomial of degree less than m. Equivalently, the point (α,β) is said to be a zero of multiplicity m of the polynomial $\mathcal{A}(X,Y)$.*

Using a well-known explicit relation (cf. [11]) between the coefficients of a bivariate polynomial $\mathcal{A}(X,Y)$ and the coefficients of the shifted polynomial, we find that Definition 2 imposes the following linear constraints

$$\sum_{i=k}^{\infty} \sum_{j=l}^{\infty} \binom{i}{k} \binom{j}{l} \alpha^{i-k} \beta^{j-l} a_{i,j} \;=\; 0 \qquad \forall\; k,l \geqslant 0 \;\text{ such that }\; k+l < m \qquad (3)$$

on the coefficients $a_{i,j}$ of $\mathcal{A}(X,Y)$. Thus $\mathcal{A}(X,Y)$ passes through a given point with multiplicity at least m if and only if its coefficients $a_{i,j}$ satisfy the $\frac{1}{2}m(m+1)$ constraints specified by (3). We are now ready to formulate the first step of the Sudan [11, 25] algorithm.

Interpolation step: *Given the set $\mathcal{P}$ of points in $\mathbb{F}_q \times \mathbb{F}_q$ and a positive integer m, compute the nontrivial bivariate polynomial $\mathcal{Q}_{\mathcal{P}}(X,Y)$ of minimal $(1,k-1)$-weighted degree that passes through all the points in $\mathcal{P}$ with multiplicity at least m.*

If $\deg_{1,k-1} \mathcal{Q}_{\mathcal{P}}(X,Y) = \delta$, then $\mathcal{Q}_{\mathcal{P}}(X,Y)$ may have up to $N_{1,k-1}(\delta)$ nonzero coefficients. These coefficients should be chosen so as to satisfy the $1/2\, nm(m+1)$ linear constraints of type (3), imposed by the interpolation step. This produces a system of $1/2\, nm(m+1)$ linear equations (not all of them necessarily linearly independent) over $\mathbb{F}_q$ in $N_{1,k-1}(\delta)$ unknowns. It is clear that this system has a solution as long as

$$N_{1,k-1}(\delta) \;>\; \frac{nm(m+1)}{2} \tag{4}$$

For efficient algorithms to solve such a system of linear equations and, hence, accomplish the interpolation step, we refer the reader to [8, 14, 18, 19, 23, 30].

The idea of Sudan's algorithm [11, 25] is that, under certain constraints on the weight of the error vector, we can read-off a list of decoding decisions as factors of $\mathcal{Q}_{\mathcal{P}}(X,Y)$ of type $Y - f(X)$. Thus the second (and last) step of the Sudan algorithm is as follows.

Factorization step: *Given the bivariate polynomial $\mathcal{Q}_{\mathcal{P}}(X,Y)$, identify all the factors of $\mathcal{Q}_{\mathcal{P}}(X,Y)$ of type $Y - f(X)$ with $\deg f(X) < k$. The output of the algorithm is a list of the codewords that correspond to these factors.*

Notice that full factorization of $\mathcal{Q}_{\mathcal{P}}(X,Y)$ is not required to find all the factors of type $Y - f(X)$ with $\deg f(X) < k$. Efficient algorithms to accomplish such "partial factorization" are given in [1, 7, 8, 18, 30]. The eventual decoder output can be taken as the codeword on the list produced at the factorization step that is closest to the received vector $\underline{y}$.

The fundamental question is under which conditions can one guarantee that the correct decoding decision is found on the list. The answer to this question is given in Theorem 2.

Theorem 2. *Suppose that a vector $\underline{y}$ and a positive integer m are given. Then the factorization step produces a list that contains all codewords of $\mathbb{C}_q(n,k)$ at distance less than*

$$t \;=\; n - \left\lfloor \frac{\delta}{m} \right\rfloor \;>\; \left\lfloor n\left(1 - \sqrt{R\,\frac{m+1}{m}}\right)\right\rfloor \tag{5}$$

from $\underline{y}$, where δ is the smallest integer such that $N_{1,k-1}(\delta) > 1/2\, nm(m+1)$ and $R = k/n$.

For a proof of (5), see [11, 18]. The inequality in (5) follows from (2). Here, we observe that Theorem 2 is a special case of Theorem 3, which we prove in the next section. Theorem 2 is the main result of Guruswami and Sudan in [11]. The theorem shows that as $m \to \infty$, the algorithm of [11] corrects any fraction of $\tau \leqslant 1 - \sqrt{R}$ erroneous positions.

3. Algebraic soft-decision decoding

In many situations [4, 27], the decoder can be supplied with probabilistic reliability information concerning the received symbols. A decoding algorithm that utilizes such information is generally referred to as a *soft-decision* decoding algorithm. We now specify this notion precisely, in the context of the present paper. First, we define a *memoryless channel*, or simply a *channel*, as a collection of a finite input alphabet $\mathscr{X}$, an output alphabet $\mathscr{Y}$, and $|\mathscr{X}|$ functions

$$f(\cdot|x) : \mathscr{Y} \to [0,1] \qquad \text{for all } x \in \mathscr{X} \tag{6}$$

that are assumed to be known to the decoder. We think of channel input and output as random variables $\mathcal{X}$ and $\mathcal{Y}$, respectively, and assume that $\mathcal{X}$ is uniformly distributed over $\mathscr{X}$. If the channel is continuous (e.g. Gaussian), then $\mathcal{Y}$ is continuous and the $f(\cdot|x)$ are probability-density functions, while if the channel is discrete then $\mathcal{Y}$ is discrete and the $f(\cdot|x)$ are probability-mass functions. In either case, the decoder can easily compute the probability that $\alpha \in \mathscr{X}$ was transmitted given that $y \in \mathscr{Y}$ was observed, as follows

$$\Pr\Big(\mathcal{X}=\alpha \mid \mathcal{Y}=y\Big) \;=\; \frac{f(y|\alpha)\Pr(\mathcal{X}=\alpha)}{\sum_{x\in\mathscr{X}} f(y|x)\Pr(\mathcal{X}=x)} \;=\; \frac{f(y|\alpha)}{\sum_{x\in\mathscr{X}} f(y|x)} \tag{7}$$

where the second equality follows from the assumption that $\mathcal{X}$ is uniform. For Reed-Solomon codes, the input alphabet is always fixed to $\mathscr{X} = \mathbb{F}_q$. Henceforth, let $\alpha_1, \alpha_2, \ldots, \alpha_q$ be a fixed ordering of the elements of $\mathbb{F}_q$; this ordering will be implicitly assumed in the remainder of this paper. Given the vector $\underline{y} = (y_1, y_2, \ldots, y_n) \in \mathscr{Y}^n$ observed at the channel output, we compute

$$\pi_{i,j} \overset{\text{def}}{=} \Pr\Big(\mathcal{X}=\alpha_i \mid \mathcal{Y}=y_j\Big) \qquad \text{for } i = 1,2,\ldots,q \text{ and } j = 1,2,\ldots,n \tag{8}$$

according to the expression in (7). Let Π be the $q \times n$ matrix with entries $\pi_{i,j}$ defined in (8). We will refer to Π as the *reliability matrix* and assume that Π is the input to a soft-decision decoding algorithm. For notational convenience, we will sometimes write $\Pi(\alpha, j)$ to refer to the entry found in the j-th column of Π in the row indexed by $\alpha \in \mathbb{F}_q$.

We note that in some applications [4, 29], it is the reliability matrix Π rather than the vector $\underline{y} \in \mathscr{Y}^n$ that is directly available at the channel output. In many other cases, the channel output alphabet $\mathscr{Y}$ is quite different from $\mathbb{F}_q$. Thus the first step in hard-decision decoding is the construction of the *hard-decision vector* $\underline{u} = (u_1, u_2, \ldots, u_n) \in \mathbb{F}_q^n$, where

$$u_j \overset{\text{def}}{=} \text{argmax}_{\alpha\in\mathbb{F}_q} \Pi(\alpha, j) \qquad \text{for } j = 1,2,\ldots,n \tag{9}$$

This hard-decision vector is then taken as the channel output $\underline{y} = \underline{c} + \underline{e}$ (cf. Section 2), thereby converting the channel at hand into a hard-decision channel.

On the other hand, a soft-decision decoder works directly with the probabilities compiled in the reliability matrix Π. If the decoder is algebraic, it must convert these probabilities into algebraic conditions. Before presenting a formal description of the proposed soft-decision decoding procedure, we give an example that illustrates the main idea.

Example 1. Let $q = 5$, so that $\mathbb{F}_q$ is the set of integers $\{0, 1, 2, 3, 4\}$ with operations modulo 5. We take $\mathcal{D} = \mathbb{F}_q = \mathbb{Z}_5$ and consider the Reed-Solomon code $\mathbb{C}_5(5, 2)$ defined as

$$\mathbb{C}_5(5,2) \overset{\text{def}}{=} \Big\{ (f(0), f(1), f(2), f(3), f(4)) \ : \ f(X) = a + bX \ \text{ with } a, b \in \mathbb{Z}_5 \Big\} \tag{10}$$

Suppose that the codeword $\underline{c} = (1, 2, 3, 4, 0)$ corresponding to $f(X) = 1 + X$ was transmitted, resulting in the following reliability matrix

$$\Pi \ = \ \begin{bmatrix} 0.01 & 0.0025 & 0.05 & 0.14 & 0.20 \\ 0.06 & 0.0025 & 0.09 & 0.14 & 0.05 \\ 0.02 & 0.9900 & 0.15 & 0.07 & 0.20 \\ 0.01 & 0.0012 & 0.61 & 0.44 & 0.40 \\ 0.90 & 0.0038 & 0.10 & 0.21 & 0.15 \end{bmatrix} \tag{11}$$

In this example, we assume, for convenience, that the rows and columns of Π are indexed by the elements $0, 1, 2, 3, 4$ of $\mathbb{Z}_5$, in this order. The hard-decision vector derived from Π according to (9) is $\underline{u} = (4, 2, 3, 3, 3)$, which corresponds to errors in positions 0, 3 and 4.

It follows that even a maximum-likelihood hard-decision decoder will fail to reconstruct the transmitted codeword $\underline{c}$, since there exists another codeword $(3, 3, 3, 3, 3) \in \mathbb{C}_5(5, 2)$ that is closer to $\underline{u}$ in the Hamming metric. The list-decoding algorithm of Guruswami-Sudan [11] will fail as well, since the number of erroneous positions exceeds the error-correction capability of the algorithm (cf. Theorem 2). The GMD soft-decision decoding algorithm [9, 10] will also fail to reconstruct $\underline{c} = (1, 2, 3, 4, 0)$. Since the last three positions in $\underline{u} = (4, 2, 3, 3, 3)$ are the least reliable, the GMD decoder will perform two decoding trials, attempting to correct $\underline{u}' = (4, 2, 3, 3, \phi)$ and $\underline{u}'' = (4, 2, \phi, \phi, \phi)$, where ϕ denotes erasure. However, the decoder will produce $(4, 2, 0, 3, 1) \in \mathbb{C}_5(5, 2)$ in both trials.

Nevertheless, we now show that the transmitted codeword can, in fact, be reconstructed without resorting to full maximum-likelihood soft-decision decoding. The idea is to select the interpolation points and their multiplicities so as to reflect the information in Π in as much as possible. A simple greedy procedure for this purpose is derived in the next section. For the special case of our example, this procedure produces the following list

point (x, y)	$(1,2)$	$(0,4)$	$(2,3)$	$(3,3)$	$(4,3)$
multiplicity	3	2	2	1	1

(12)

These points and multiplicities are shown to be optimal for the reliability matrix in (11), in a precise sense described in Section 4. The minimal (1, 1)-weighted degree polynomial that passes through all the points in (12) with the required multiplicities turns out to be

$$\begin{aligned} \mathcal{Q}(X,Y) &= 1 + X + Y - X^2 - Y^2 - 2X^2Y + Y^2X - Y^3 + X^4 - 2YX^3 - X^2Y^2 + 2Y^3X \\ &= (Y - X - 1)(Y - 3X - 4)(1 + Y + 3X + 3X^2 + 3XY) \end{aligned}$$

We identify the two solutions $f_1(X) = 1 + X$ and $f_2(X) = 4 + 3X$ as corresponding to $(1, 2, 3, 4, 0)$ and $(4, 2, 0, 3, 1)$, respectively. Referring to the reliability matrix in (11), we see that $\Pi(1,0)\,\Pi(2,1)\,\Pi(3,2)\,\Pi(4,3)\,\Pi(0,4) > \Pi(4,0)\,\Pi(2,1)\,\Pi(0,2)\,\Pi(3,3)\,\Pi(1,4)$. Thus the transmitted codeword $\underline{c} = (1, 2, 3, 4, 0)$ is more likely than $(4, 2, 0, 3, 1)$ given the observations; it will therefore be selected as the decoder output. $\square$

Example 1 shows how a soft-decision decoding algorithm for Reed-Solomon codes might work: the "soft" reliability information enters the decoding process through the choice of interpolation points and their multiplicities.

A convenient way to keep track of the interpolation points and their multiplicities is by means of a multiplicity matrix. A *multiplicity matrix* is a $q \times n$ matrix M with nonnegative integer entries $m_{i,j}$. The first step of our soft-decision decoding algorithm consists of computing the multiplicity matrix M from the reliability matrix Π. This step is discussed in detail in the next section. In the remainder of this section, we characterize the proposed decoding algorithm for a *given choice* of interpolation points and their multiplicities. Thus the second step is the "soft" interpolation step, which may be expressed as follows.

Soft interpolation step: *Given the point set $\mathcal{D} = \{x_1, x_2, \ldots, x_n\}$ and the multiplicity matrix $M = [m_{i,j}]$, compute the (nontrivial) bivariate polynomial $\mathcal{Q}_M(X,Y)$ of minimal $(1,k{-}1)$-weighted degree that has a zero of multiplicity at least $m_{i,j}$ at the point (x_j, α_i) for every i,j such that $m_{i,j} \neq 0$.*

The third and final step of our algorithm is the factorization step, which is identical to the factorization step of the Sudan algorithm, described in the previous section. (As in Example 1, the soft-decision list-decoder may also include a post-processor that selects the most likely codeword from the list produced at the factorization step.)

Definition 3. *Given a $q \times n$ matrix M with nonnegative integer entries $m_{i,j}$, we define the cost of M as follows*

$$\mathcal{C}(M) \ \stackrel{\text{def}}{=}\ \frac{1}{2}\sum_{i=1}^{q}\sum_{j=1}^{n} m_{i,j}(m_{i,j}+1)$$

It is easy to see that the computation of the polynomial $\mathcal{Q}_M(X,Y)$ is equivalent to solving a system of linear equations of type (3). Since a given zero of multiplicity m imposes $\frac{1}{2}m(m+1)$ linear constraints on the coefficients of $\mathcal{Q}_M(X,Y)$, the cost $\mathcal{C}(M)$ is precisely the total number of linear equations. As in (4), we can always find a solution $\mathcal{Q}_M(X,Y)$ to the soft interpolation task if the $(1,k{-}1)$-weighted degree δ is large enough, namely if

$$N_{1,k-1}(\delta) \ > \ \mathcal{C}(M) \tag{13}$$

so that the number of degrees of freedom is greater than the number of linear constraints. Thus we define the function

$$\Delta_{w_X,w_Y}(\nu) \ \stackrel{\text{def}}{=}\ \min\,\{\,\delta \in \mathbb{Z} \ : \ N_{w_X,w_Y}(\delta) > \nu\,\} \tag{14}$$

Notice that $\Delta_{1,k-1}(\nu) < \sqrt{2(k{-}1)\nu} < \sqrt{2k\nu}$ in view of (2). Next, given two $q \times n$ matrices A and B over the same field, we define the inner product

$$\langle A, B\rangle \ \stackrel{\text{def}}{=}\ \mathsf{trace}(AB^T) \ = \ \sum_{i=1}^{q}\sum_{j=1}^{n} a_{i,j}b_{i,j}$$

Finally, it will be convenient to think of the codewords of the Reed-Solomon code $\mathbb{C}_q(n,k)$ as $q \times n$ matrices over the reals. Specifically, any vector $\underline{v} = (v_1, v_2, \ldots, v_n)$ over $\mathbb{F}_q$ can be represented by the $q \times n$ real-valued matrix $[\underline{v}]$ defined as follows: $[\underline{v}]_{i,j} = 1$ if $v_j = \alpha_i$, and $[\underline{v}]_{i,j} = 0$ otherwise. With this notation, we have the following definition.

Definition 4. *The score of a vector $\underline{v} = (v_1, v_2, \ldots, v_n)$ over $\mathbb{F}_q$ with respect to a given multiplicity matrix M is defined as the inner product $\mathcal{S}_M(\underline{v}) = \langle M, [\underline{v}] \rangle$.*

The following theorem characterizes the set of codewords produced by our soft-decision list-decoding algorithm for a given multiplicity matrix. Notice that Theorem 2 follows as a special case of Theorem 3 for the multiplicity matrix $M = m\,[\underline{y}]$.

Theorem 3. *Let $\mathcal{C} = \mathcal{C}(M)$ be the cost of a given multiplicity matrix M. Then the polynomial $\mathcal{Q}_M(X,Y)$ has a factor $Y - f(X)$, where $f(X)$ evaluates to a codeword $\underline{c} \in \mathbb{C}_q(n,k)$, if the score of $\underline{c}$ is large enough compared to $\mathcal{C}$, namely if*

$$\mathcal{S}_M(\underline{c}) \;>\; \Delta_{1,k-1}(\mathcal{C}) \tag{15}$$

Proof. Let $\underline{c} = (c_1, c_2, \ldots, c_n)$ be a codeword of $\mathbb{C}_q(n,k)$, and let $f(X)$ be the polynomial that evaluates to $\underline{c}$. That is $f(x_j) = c_j$ for all $x_j \in \mathcal{D}$, where $\mathcal{D}$ is the set of points that define $\mathbb{C}_q(n,k)$ as in (1). Given $\mathcal{Q}_M(X,Y)$, we define the polynomial $g(X) \in \mathbb{F}_q[X]$ as follows

$$g(X) \;\overset{\text{def}}{=}\; \mathcal{Q}_M(X, f(X))$$

It would clearly suffice to prove that (15) implies that $g(X)$ is the all-zero polynomial, since then $\mathcal{Q}_M(X,Y)$ must be divisible by $Y - f(X)$. To prove that $g(X) \equiv 0$, we will show that $\deg g(X) \leqslant \Delta_{1,k-1}(\mathcal{C})$ and yet $g(X)$ has a factor of degree $\mathcal{S}_M(\underline{c})$. We write

$$\mathcal{S}_M(\underline{c}) \;=\; \langle M, [\underline{c}] \rangle \;=\; m_1 + m_2 + \cdots + m_n$$

Thus the polynomial $\mathcal{Q}_M(X,Y)$ passes through the point (x_j, c_j) with multiplicity at least m_j, for $j = 1, 2, \ldots, n$. We will make use of the following lemma.

Lemma 4. *Suppose that a bivariate polynomial $\mathcal{Q}(X,Y)$ passes through a point (α, β) in $\mathbb{F}_q \times \mathbb{F}_q$ with multiplicity at least m, and let $p(X)$ be any polynomial in $\mathbb{F}_q[X]$ such that $p(\alpha) = \beta$. Then the polynomial $\mathcal{Q}(X, p(X))$ is divisible by $(X - \alpha)^m$.*

This lemma is identical to Lemma 4 of [11], and we omit the proof. Since $f(x_j) = c_j$ for $j = 1, 2, \ldots, n$, it follows from Lemma 4 and the fact that $x_1, x_2, \ldots, x_n$ are all distinct that the polynomial $g(X) = \mathcal{Q}_M(X, f(X))$ is divisible by the product

$$(X - x_1)^{m_1} (X - x_2)^{m_2} \cdots (X - x_n)^{m_n}$$

whose degree is $\mathcal{S}_M(\underline{c})$. We conclude that either $\deg g(X) \geqslant \mathcal{S}_M(\underline{c})$ or $g(X) \equiv 0$. Since $\deg f(X) \leqslant k - 1$, it is easy to see that the degree of $g(X) = \mathcal{Q}_M(X, f(X))$ cannot exceed the $(1, k{-}1)$-weighted degree of $\mathcal{Q}_M(X,Y)$. Yet it follows from (13) and (14) that $\deg_{1,k-1} \mathcal{Q}_M(X,Y) \leqslant \Delta_{1,k-1}(\mathcal{C})$. Thus if $g(X) \not\equiv 0$ then $\mathcal{S}_M(\underline{c}) \leqslant \deg g(X) \leqslant \Delta_{1,k-1}(\mathcal{C})$. Hence (15) implies that $g(X) \equiv 0$. ∎

Corollary 5. *Let $\mathcal{C} = \mathcal{C}(M)$ be the cost of a given multiplicity matrix. Then $\mathcal{Q}_M(X,Y)$ has a factor $Y - f(X)$, where $f(X)$ evaluates to $\underline{c} \in \mathbb{C}_q(n,k)$, if $\mathcal{S}_M(\underline{c}) \geqslant \sqrt{2(k{-}1)\mathcal{C}}$.*

Proof. Follows from Theorem 3 and the fact that $\Delta_{1,k-1}(\mathcal{C}) < \sqrt{2(k{-}1)\mathcal{C}}$ by Lemma 1. ∎

4. From posterior probabilities to interpolation points

This section deals with the conversion of posterior probabilities derived from the channel output into a choice of interpolation points and their multiplicities. More specifically, given a reliability matrix Π, as defined in (8), we would like to compute the multiplicity matrix M that serves as the input to the soft interpolation step of our algorithm.

Let $\mathscr{M}_{q,n}$ denote the set of all $q \times n$ matrices with nonnegative integer entries $m_{i,j}$, and let $\mathscr{M}(\mathcal{C})$ be the finite set of all matrices in $\mathscr{M}_{q,n}$ whose cost is equal to $\mathcal{C}$. Thus

$$\mathscr{M}(\mathcal{C}) \stackrel{\text{def}}{=} \left\{ M \in \mathscr{M}_{q,n} \ : \ \frac{1}{2}\sum_{i=1}^{q}\sum_{j=1}^{n} m_{i,j}(m_{i,j}{+}1) = \mathcal{C} \right\}$$

In view of Theorem 3, we would like to choose $M \in \mathscr{M}(\mathcal{C})$ so as to maximize the score of the transmitted codeword $\underline{c} \in \mathbb{C}_q(n,k)$. However, the transmitted codeword itself is obviously unknown to the decoder; only some stochastic information about $\underline{c}$ is available through the observation of the channel output $(y_1, y_2, \ldots, y_n) \in \mathscr{Y}^n$ and the knowledge of the channel transition probabilities $\Pr(\mathcal{X} = \alpha \,|\, \mathcal{Y} = y)$. In fact, as far as the decoder is concerned, the transmitted codeword may be thought of as a random vector, which we denote by $\boldsymbol{\mathcal{X}} = (\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n)$. For a given multiplicity matrix M, the score of the transmitted codeword is a function of $\boldsymbol{\mathcal{X}}$ given by $\mathcal{S}_M(\boldsymbol{\mathcal{X}}) = \langle M, [\boldsymbol{\mathcal{X}}] \rangle$.

Thus $\mathcal{S}_M(\boldsymbol{\mathcal{X}})$ is a random variable, and the question is: what is the best choice of a multiplicity matrix $M \in \mathscr{M}(\mathcal{C})$ in this probabilistic setting? We choose to compute the matrix $M \in \mathscr{M}(\mathcal{C})$ that maximizes the expected value of $\mathcal{S}_M(\boldsymbol{\mathcal{X}})$. This choice is based on the following considerations. First, this is a reasonable optimization criterion for the probabilistic setup which is the focus of this paper. The obvious alternative is to compute $M \in \mathscr{M}(\mathcal{C})$ that maximizes the probability that $\mathcal{S}_M(\boldsymbol{\mathcal{X}}) > \Delta(\mathcal{C})$. However, this computation appears to be extremely difficult, except for certain special cases of simple channels.

The second reason is this: Theorem 14 of Section 5.2 shows that this criterion is asymptotically optimal in the following sense. Let P_e denote the probability of list-decoding failure, defined as the probability that the transmitted codeword $\underline{c}$ is not on the list produced by the soft-decoder. Theorem 14 implies that for every $\varepsilon > 0$, however small, we have

$$\sqrt{R} \ \leqslant \ \frac{\mathsf{E}\{\mathcal{S}_M(\boldsymbol{\mathcal{X}})\}}{\sqrt{2n\mathcal{C}}} - \frac{1}{\sqrt{\varepsilon n}} \qquad \Rightarrow \qquad P_e \leqslant \varepsilon \tag{16}$$

where $R = k/n$ is the rate of the Reed-Solomon code and $\mathsf{E}\{\mathcal{S}_M(\boldsymbol{\mathcal{X}})\}$ is the expected value of the score for a given multiplicity matrix $M \in \mathscr{M}(\mathcal{C})$. On the other hand, under certain assumptions stated in Section 5.2, for every $0 < \varepsilon < 1$ we have

$$P_e \leqslant \varepsilon \qquad \Rightarrow \qquad \sqrt{R - {}^{1}\!/_{n}} \ \leqslant \ \frac{\mathsf{E}\{\mathcal{S}_M(\boldsymbol{\mathcal{X}})\}}{\sqrt{2n\mathcal{C}}} + \frac{1}{\sqrt{(1{-}\varepsilon)n}} \tag{17}$$

It is easy to see that for $n \to \infty$, the two bounds on R coincide. Thus, at least asymptotically, maximizing the expectation of the score allows for reliable transmission at the highest possible rate. Of course, one might argue that such asymptotic reasoning has little meaning for Reed-Solomon codes, since $n \leqslant q$. However, the proposed soft-decoding algorithm can be generalized to algebraic-geometric codes, so that $n \to \infty$ makes sense for a fixed q. More importantly, the bounds in (17) and (16) essentially follow from the fact that the random variable $\mathcal{S}_M(\boldsymbol{\mathcal{X}})$ concentrates about its expected value as n becomes large. We have observed that in practice (in simulations), the length n does not have to be too large for this concentration to take place. In fact, for signal-to-noise ratios (SNRs) of practical interest, $n = 256$ is usually enough.

To proceed, let us define the ***expected score*** with respect to a probability distribution $P(\cdot)$ on the random vector $\boldsymbol{\mathcal{X}} = (\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n)$ as follows

$$\mathsf{E}_P\{\mathcal{S}_M(\boldsymbol{\mathcal{X}})\} \overset{\text{def}}{=} \sum_{\underline{x} \in \mathscr{X}^n} \mathcal{S}_M(\underline{x}) P(\underline{x}) = \sum_{\underline{x} \in \mathbb{F}_q^n} \sum_{j=1}^{n} M(x_j, j) P(\underline{x})$$

where $M(x_j, j)$ denotes the entry found in the j-th column of M in the row indexed by x_j. It remains to specify $P(\cdot)$. For this purpose, we adopt the product distribution determined by the channel output $(y_1, y_2, \ldots, y_n) \in \mathscr{Y}^n$, namely

$$P(x_1, x_2, \ldots, x_n) \overset{\text{def}}{=} \prod_{j=1}^{n} \Pr\Big(\mathcal{X}_j = x_j \mid \mathcal{Y}_j = y_j\Big) = \prod_{j=1}^{n} \Pi(x_j, j) \tag{18}$$

where Π is the reliability matrix defined in (8). It is easy to see that this would be the *a posteriori* distribution of $\boldsymbol{\mathcal{X}}$ given the channel observations, if the *a priori* distribution of $\boldsymbol{\mathcal{X}}$ were uniform over the space $\mathbb{F}_q^n$. However, the decoder knows that $\boldsymbol{\mathcal{X}}$ was drawn *a priori* from the code $\mathbb{C}_q(n,k)$ rather than from the entire space $\mathbb{F}_q^n$, and hence the probability model in (18) is suboptimal. Taking this into account results in the probability model given in (50). This model is optimal in that it reflects precisely *all* the information available to the decoder. Unfortunately, this model leads to an intractable optimization problem, as shown in Appendix A (cf. Theorem 18). Thus the remainder of this section is concerned with the computation of the matrix $M(\Pi, \mathcal{C})$ defined as follows

$$M(\Pi, \mathcal{C}) \overset{\text{def}}{=} \operatorname{argmax}_{M \in \mathscr{M}(\mathcal{C})} \mathsf{E}_P\{\mathcal{S}_M(\boldsymbol{\mathcal{X}})\} \tag{19}$$

where the expectation is taken with respect to the probability distribution $P(\cdot)$ in (18). We start with the following lemma, which gives a useful expression for the expected score.

Lemma 6. *The expected score with respect to the probability distribution in* (18) *is equal to the inner product of the multiplicity matrix and the reliability matrix, namely*

$$\mathsf{E}_P\{\mathcal{S}_M(\boldsymbol{\mathcal{X}})\} = \langle M, \Pi \rangle$$

Proof. Loosely speaking, the lemma follows from the fact that if $\boldsymbol{\mathcal{X}}$ is distributed according to (18), then the reliability matrix Π is precisely the expected value of $[\boldsymbol{\mathcal{X}}]$. To see this, consider the random vector $\boldsymbol{\mathcal{X}}' = (\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_{n-1})$ obtained by deleting the last component of $\boldsymbol{\mathcal{X}} = (\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n)$. The probability distribution of $\boldsymbol{\mathcal{X}}'$ can be computed by marginalizing (18) with respect to $\mathcal{X}_n$. Explicitly, if $\underline{x}' = (x_1, x_2, \ldots, x_{n-1}) \in \mathbb{F}_q^{n-1}$ then

$$\Pr\Big(\boldsymbol{\mathcal{X}}' = \underline{x}'\Big) \;=\; \sum_{x_n \in \mathbb{F}_q} P(x_1, x_2, \ldots, x_{n-1}, x_n) \;=\; \sum_{x_n \in \mathbb{F}_q} \prod_{j=1}^{n} \Pi(x_j, j) \;=\; \prod_{j=1}^{n-1} \Pi(x_j, j)$$

where the last equality follows from the fact that $\sum_{x_n \in \mathbb{F}_q} \Pi(x_n, n) = 1$. This, in particular, implies that

$$1 \;=\; \sum_{\underline{x}' \in \mathbb{F}_q^{n-1}} \Pr\Big(\boldsymbol{\mathcal{X}}' = \underline{x}'\Big) \;=\; \sum_{\underline{x}' \in \mathbb{F}_q^{n-1}} \prod_{j=1}^{n-1} \Pi(x_j, j) \;=\; \sum_{\substack{\underline{x} \in \mathbb{F}_q^n \\ x_j = \alpha}} \prod_{\substack{l=1 \\ l \neq j}}^{n} \Pi(x_l, l) \tag{20}$$

for any $j \in \{1, 2, \ldots, n\}$ and any $\alpha \in \mathbb{F}_q$. The last equality in (20) follows by applying a similar argument to the random vector obtained by deleting the j-th component of $\boldsymbol{\mathcal{X}}$. Now consider the $q \times n$ matrix $\mathscr{P} = [p_{i,j}]$ which may be thought of as the expected value of $[\boldsymbol{\mathcal{X}}]$ with respect to the distribution $P(\cdot)$ in (18). Specifically, we define $\mathscr{P}$ as follows

$$\mathscr{P} \;\stackrel{\text{def}}{=}\; \sum_{\underline{x} \in \mathbb{F}_q^n} [\underline{x}] P(\underline{x}) \tag{21}$$

Since $[\underline{x}]_{i,j} = 1$ if $x_j = \alpha_i$, and $[\underline{x}]_{i,j} = 0$ otherwise, the entry found in row i and column j of the matrix $\mathscr{P}$ is given by

$$p_{i,j} \;=\; \sum_{\substack{\underline{x} \in \mathbb{F}_q^n \\ x_j = \alpha_i}} P(\underline{x}) \;=\; \sum_{\substack{\underline{x} \in \mathbb{F}_q^n \\ x_j = \alpha_i}} \prod_{l=1}^{n} \Pi(x_l, l) \;=\; \Pi(\alpha_i, j) \sum_{\substack{\underline{x} \in \mathbb{F}_q^n \\ x_j = \alpha_i}} \prod_{\substack{l=1 \\ l \neq j}}^{n} \Pi(x_l, l) \tag{22}$$

The summation on the right-hand side of (22) evaluates to 1 by (20), which implies that $p_{i,j} = \Pi(\alpha_i, j) = \pi_{i,j}$ for all $i \in \{1, 2, \ldots, q\}$ and all $j \in \{1, 2, \ldots, n\}$. Therefore $\mathscr{P} = \Pi$. The lemma can be now easily proved by interchanging expectation with inner product $\mathsf{E}_P\{\mathcal{S}_M(\boldsymbol{\mathcal{X}})\} = \mathsf{E}_P\{\langle M, [\boldsymbol{\mathcal{X}}]\rangle\} = \langle M, \mathsf{E}_P\{[\boldsymbol{\mathcal{X}}]\}\rangle = \langle M, \Pi\rangle$. More explicitly, we have

$$\mathsf{E}_P\{\mathcal{S}_M(\boldsymbol{\mathcal{X}})\} \;=\; \sum_{\underline{x} \in \mathbb{F}_q^n} \langle M, [\underline{x}]\rangle\, P(\underline{x}) \;=\; \sum_{\underline{x} \in \mathbb{F}_q^n} \langle M, [\underline{x}] P(\underline{x})\rangle \;=\; \Big\langle M, \sum_{\underline{x} \in \mathbb{F}_q^n} [\underline{x}] P(\underline{x}) \Big\rangle \;=\; \langle M, \Pi\rangle$$

where the first two equalities follow from the linearity of the inner product, while the last equality follows from the definition of $\mathscr{P} = \Pi$ in (21). ∎

We will construct $M(\Pi, \mathcal{C})$ iteratively, starting with the all-zero matrix and increasing one of the entries in the matrix at each iteration. Referring to Lemma 6, we see that increasing $m_{i,j}$ from 0 to 1 increases the expected score by $\pi_{i,j}$ while increasing the cost by 1. If we require that $\mathcal{Q}_M(X, Y)$ passes through the same point again — that is, increase $m_{i,j}$ from 1 to 2 — then the expected score again grows by $\pi_{i,j}$, but now we have to "pay" two additional linear constraints. In general, increasing $m_{i,j}$ from a to $a+1$ always increases the expected score by $\pi_{i,j}$ while introducing $a+1$ additional constraints of type (3).

Example 2. Returning to Example 1 of the previous section, consider the code $\mathbb{C}_5(5,2)$ given by (10) and the reliability matrix Π in (11). Suppose we restrict the cost of the multiplicity matrix to 14, that is, we wish to find

$$M(\Pi, 14) \;=\; \text{argmax}_{M \in \mathscr{M}(14)} \langle M, \Pi \rangle$$

We construct a multiplicity matrix M by a greedy iterative process, starting with the 5×5 all-zero matrix, and requiring at each iteration that the newly chosen interpolation point maximizes the increase in the expected score normalized by the number of additional linear constraints (the increase in cost).

Table 1 shows the sequence of chosen interpolation points. Observe that the column that contains the ratio of the increase in the expected score to the increase in cost is strictly decreasing. The resulting multiplicity matrix M is described in equation (12) of Example 1. It can be verified by exhaustive search that $\max_{M \in \mathscr{M}(14)} \langle M, \Pi \rangle = 6.83$, so $M = M(\Pi, 14)$.

Notice that $N_{1,1}(3) = 10$ while $N_{1,1}(4) = 15$ by Lemma 1, so that $\Delta_{1,1}(14) = 4$. Thus the *expected* score exceeds the minimum score required for successful decoding (cf. Theorem 3) by a factor of about 1.7. This gives a high level of confidence that the *actual* score of the transmitted codeword will also exceed $\Delta_{1,k-1}(14) = 4$. Indeed, the score of $\underline{c} = (1,2,3,4,0) \in \mathbb{C}_5(5,2)$ with respect to M is $\mathcal{S}_M(\underline{c}) = 5$. $\square$

The greedy iterative procedure used in Example 2 turned out to be optimal for that case. We formalize this procedure as Algorithm A below.

Algorithm A

Input: Reliability matrix Π and a positive integer s, indicating the total number of interpolation points.

Output: Multiplicity matrix M.

Initialization step: Set $\Pi^* := \Pi$ and $M :=$ `all-zero matrix`.

Iteration step: Find the position (i,j) of the largest entry $\pi^*_{i,j}$ in Π^*, and set

$$\begin{aligned} \pi^*_{i,j} &:= \frac{\pi_{i,j}}{m_{i,j}+2} \\ m_{i,j} &:= m_{i,j}+1 \\ s &:= s-1 \end{aligned}$$

Control step: If $s = 0$, return M; otherwise go to the iteration step.

Let $\mathcal{M}(\Pi, s)$ denote the multiplicity matrix produced by Algorithm A for a given reliability matrix Π and a given number of interpolation points s (counted with multiplicities). The following theorem shows that this matrix is optimal.

itera-tion	interpolation point (i,j)	increase in score	increase in cost	$\frac{d\,\text{score}}{d\,\text{cost}}$	total cost	expected score
1	(1,2)	0.99	1	0.990	1	0.99
2	(0,4)	0.90	1	0.900	2	1.89
3	(2,3)	0.61	1	0.610	3	2.50
4	(1,2)	0.99	2	0.495	5	3.49
5	(0,4)	0.90	2	0.450	7	4.39
6	(3,3)	0.44	1	0.440	8	4.83
7	(4,3)	0.40	1	0.400	9	5.23
8	(1,2)	0.99	3	0.330	12	6.22
9	(2,3)	0.61	2	0.305	14	6.83

Table 1. *Iterative construction of a multiplicity matrix*

Theorem 7. *The matrix $\mathcal{M}(\Pi, s)$ maximizes the expected score among all matrices in $\mathscr{M}_{q,n}$ with the same cost. That is, if $\mathcal{C}$ is the cost of $\mathcal{M}(\Pi, s)$, then*

$$\mathcal{M}(\Pi, s) \;=\; \text{argmax}_{M\in\mathscr{M}(\mathcal{C})} \langle M, \Pi\rangle$$

Proof. With each position (i,j) in the reliability matrix Π, we associate an infinite sequence of rectangles $\mathcal{B}_{i,j,1}, \mathcal{B}_{i,j,2}, \ldots$ indexed by the positive integers. Let $\mathfrak{B}$ denote the set of all such rectangles. For each rectangle $\mathcal{B}_{i,j,l} \in \mathfrak{B}$, we define $\texttt{length}(\mathcal{B}_{i,j,l}) = l$, $\texttt{height}(\mathcal{B}_{i,j,l}) = \pi_{i,j}/l$, and $\texttt{area}(\mathcal{B}_{i,j,l}) = \texttt{length}(\mathcal{B}_{i,j,l}) \cdot \texttt{height}(\mathcal{B}_{i,j,l}) = \pi_{i,j}$. For a multiplicity matrix $M \in \mathscr{M}_{q,n}$, we define the corresponding set of rectangles $\mathscr{S}(M)$ as

$$\mathscr{S}(M) \overset{\text{def}}{=} \{\mathcal{B}_{i,j,l} \;:\; 1 \leqslant i \leqslant q,\; 1 \leqslant j \leqslant n,\; \text{and } 1 \leqslant l \leqslant m_{i,j}\} \tag{23}$$

Observe that the number of rectangles in $\mathscr{S}(M)$ is $\sum_{i=1}^{q}\sum_{j=1}^{n} m_{i,j}$, which is precisely the total number of interpolation points imposed by the multiplicity matrix M (counted with multiplicities). Furthermore

$$\mathcal{C}(M) = \sum_{\substack{i=1\\j=1}}^{q,n} \frac{m_{i,j}(m_{i,j}+1)}{2} = \sum_{\substack{i=1\\j=1}}^{q,n}\sum_{l=1}^{m_{i,j}} l = \sum_{\substack{i=1\\j=1}}^{q,n}\sum_{l=1}^{m_{i,j}} \texttt{length}(\mathcal{B}_{i,j,l}) = \sum_{\mathcal{B}\in\mathscr{S}(M)} \texttt{length}(\mathcal{B})$$

$$\langle M, \Pi\rangle = \sum_{\substack{i=1\\j=1}}^{q,n} m_{i,j}\cdot\pi_{i,j} = \sum_{\substack{i=1\\j=1}}^{q,n}\sum_{l=1}^{m_{i,j}} \pi_{i,j} = \sum_{\substack{i=1\\j=1}}^{q,n}\sum_{l=1}^{m_{i,j}} \texttt{area}(\mathcal{B}_{i,j,l}) = \sum_{\mathcal{B}\in\mathscr{S}(M)} \texttt{area}(\mathcal{B})$$

Thus the cost of M is the total length of all the rectangles in $\mathscr{S}(M)$ and the expected score $\langle M, \Pi\rangle$ is the total area of all the rectangles in $\mathscr{S}(M)$. It is intuitively clear that to maximize the total area for a given total length, one has to choose the highest rectangles. This is precisely what Algorithm A does: the algorithm constructs the matrix $\mathcal{M}(\Pi, s)$ that corresponds to the set of s highest rectangles in $\mathfrak{B}$. It is now obvious that if the s *highest* rectangles in $\mathfrak{B}$ have total length $\mathcal{C}$, then no collection of rectangles of total length at most $\mathcal{C}$ can have a larger total area. ∎

Although Algorithm A produces an optimal multiplicity matrix $\mathcal{M}(\Pi, s)$ for an arbitrary number of interpolation points s, it cannot be used to solve the optimization problem (19) for an arbitrary value of the cost $\mathcal{C}$. The algorithm computes a solution to (19) only for those costs that are expressible as the total length of the s highest rectangles in $\mathfrak{B}$ for some s. In other words, if $M(\Pi,1), M(\Pi,2), M(\Pi,3), \ldots$ is the infinite sequence of matrices defined by (19) for $\mathcal{C} = 1, 2, 3, \ldots$, then $\mathcal{M}(\Pi,1), \mathcal{M}(\Pi,2), \mathcal{M}(\Pi,3), \ldots$ is a subsequence of this sequence. This subsequence will generally suffice for our purposes.

Remark. Algorithm A can be also used to generate a sequence of multiplicity matrices indexed by a bound on the size of the list produced by the soft-decision decoder. Clearly, the number of factors of $\mathcal{Q}_M(X,Y)$ of type $Y - f(X)$ is bounded by $\deg_{0,1}\mathcal{Q}_M(X,Y)$, and

$$\deg_{0,1}\mathcal{Q}_M(X,Y) \;\leqslant\; \left\lfloor \frac{\deg_{1,k-1}\mathcal{Q}_M(X,Y)}{k-1} \right\rfloor \;\leqslant\; \left\lfloor \frac{\Delta_{1,k-1}(\mathcal{C})}{k-1} \right\rfloor \tag{24}$$

Thus, given a bound on the desired list-size, all one has to do is to keep track of the total cost $\mathcal{C}$, and stop Algorithm A just before the right-hand side of (24) exceeds this bound.

5. Asymptotic performance analysis

In the next subsection, we investigate the multiplicity matrix $\mathcal{M}(\Pi, s)$ produced by Algorithm A as $s \to \infty$. We shall see that for $s \to \infty$ this matrix becomes proportional to Π. Based on this result, we derive an asymptotic condition for successful list-decoding, and provide a geometric characterization of the asymptotic decoding regions of our algorithm. In a subsequent subsection, we focus instead on long codes — that is, we study the limiting performance of our algorithm as the code length n approaches infinity.

5.1. Asymptotic analysis for large costs

We start with two simple lemmas. In all of the subsequent analysis, we keep the reliability matrix $\Pi = [\pi_{i,j}]$ fixed, while s ranges over the positive integers. For convenience, we define $\Phi = \{1, 2, \ldots, q\} \times \{1, 2, \ldots, n\}$. Let $\chi(\Pi)$ denote the set of all $(i,j) \in \Phi$ such that $\pi_{i,j} \neq 0$. Let $m_{i,j}(s)$ denote the entries in the matrix $\mathcal{M}(\Pi, s)$ produced by Algorithm A.

Lemma 8. *As $s \to \infty$, every nonzero entry in $\mathcal{M}(\Pi, s)$ grows without bound. In other words $m_{i,j}(s) \to \infty$ when $s \to \infty$ for all $(i,j) \in \chi(\Pi)$.*

Proof. Define $m_{\max}(s) = \max_{(i,j)\in\Phi} m_{i,j}(s)$ and $m_{\min}(s) = \min_{(i,j)\in\chi(\Pi)} m_{i,j}(s)$. Clearly, it would suffice to show that $m_{\min}(s) \to \infty$ for $s \to \infty$. Notice that

$$s \;=\; \sum_{i=1}^{q}\sum_{j=1}^{n} m_{i,j}(s) \;\leqslant\; qn\, m_{\max}(s) \tag{25}$$

It follows from (25) that $m_{\max}(s) \to \infty$ as $s \to \infty$. Hence there exists an infinite integer sequence $s_1, s_2, s_3, \ldots$ defined by the property that $m_{\max}(s_r) = r$ and $m_{\max}(s_r+1) = r+1$. The iterative nature of Algorithm A implies that for all $s \geqslant 1$, there is exactly one position

(i_0, j_0) such that $m_{i_0,j_0}(s+1) = m_{i_0,j_0}(s) + 1$. We say that (i_0, j_0) is the position ***updated at iteration*** $s+1$ of Algorithm A. This position is distinguished by the property that

$$\frac{\pi_{i_0,j_0}}{m_{i_0,j_0}(s)+1} \geqslant \frac{\pi_{i,j}}{m_{i,j}(s)+1} \qquad \text{for all } (i,j) \in \Phi \tag{26}$$

For $r = 1, 2, \ldots$, let (i_r, j_r) denote the position updated at iteration $s_r + 1$ of Algorithm A. Then it follows from (26) and the definition of s_r that

$$m_{i,j}(s_r) + 1 \geqslant \frac{\pi_{i,j}}{\pi_{i_r,j_r}}\Big(m_{\max}(s_r)+1\Big) \geqslant \frac{\pi_{\min}}{\pi_{\max}} r + \frac{\pi_{\min}}{\pi_{\max}} \qquad \text{for all } (i,j) \in \chi(\Pi)$$

where $\pi_{\max} = \max_{(i,j)\in\Phi} \pi_{i,j}$ and $\pi_{\min} = \min_{(i,j)\in\chi(\Pi)} \pi_{i,j}$. Denoting by ρ the ratio $\pi_{\min}/\pi_{\max}$, we conclude from the above that $m_{\min}(s_r) \geqslant \rho r + \rho - 1$. Since ρ is a positive constant while $r \to \infty$ as $s \to \infty$, it follows that $m_{\min}(s)$ grows without bound for $s \to \infty$. ▮

Henceforth, let (i_s, j_s) denote the position updated at iteration s of Algorithm A, and consider the sequence of ratios of the increase in the expected score to the increase in cost at successive iterations of Algorithm A (cf. the fourth column of Table 1), namely

$$\theta_1 \overset{\text{def}}{=} \frac{\pi_{i_1,j_1}}{m_{i_1,j_1}(1)}, \quad \theta_2 \overset{\text{def}}{=} \frac{\pi_{i_2,j_2}}{m_{i_2,j_2}(2)}, \quad \theta_3 \overset{\text{def}}{=} \frac{\pi_{i_3,j_3}}{m_{i_3,j_3}(3)}, \quad \cdots$$

It follows from (26) that the sequence $\theta_1, \theta_2, \ldots$ is non-increasing. (Indeed, this was our goal in the design of Algorithm A.) Clearly $\theta_1 = \pi_{\max}$ while $\lim_{s\to\infty} \theta_s = 0$ by Lemma 8.

Lemma 9. *For every positive integer s, there exists a positive constant $K = K(s) \leqslant \pi_{\max}$, such that*

$$K(m_{i,j}(s)+1) \geqslant \pi_{i,j} \geqslant K m_{i,j}(s) \qquad \textit{for all } (i,j) \in \Phi \tag{27}$$

Conversely, for every positive constant $K \leqslant \pi_{\max}$, there exists a positive integer $s = s(K)$ such that (27) *holds.*

Proof. Given s, we choose $K = K(s)$ so that $\theta_{s+1} \leqslant K \leqslant \theta_s$, which is always possible as the sequence $\theta_1, \theta_2, \ldots$ is non-increasing. To prove the first inequality in (27), observe that

$$K \geqslant \theta_{s+1} = \frac{\pi_{i_{s+1},j_{s+1}}}{m_{i_{s+1},j_{s+1}}(s+1)} = \frac{\pi_{i_{s+1},j_{s+1}}}{m_{i_{s+1},j_{s+1}}(s)+1} \geqslant \frac{\pi_{i,j}}{m_{i,j}(s)+1} \qquad \text{for all } (i,j) \in \Phi$$

where the last inequality follows from (26). The second inequality in (27) holds vacuously if $m_{i,j}(s) = 0$, so assume that $m_{i,j}(s) \geqslant 1$. This assumption implies that position (i,j) was updated at least once, and we let $s^* \leqslant s$ denote the number of the *most recent* iteration of Algorithm A at which position (i,j) was updated. Then

$$K \leqslant \theta_s \leqslant \theta_{s^*} = \frac{\pi_{i,j}}{m_{i,j}(s^*)} = \frac{\pi_{i,j}}{m_{i,j}(s)}$$

where the last equality follows from the fact that position (i,j) was *not* updated since iteration s^*. Finally, given $0 < K \leqslant \pi_{\max}$, we choose $s = s(K)$ so that $\theta_{s+1} < K \leqslant \theta_s$ once again. This choice is possible because the sequence $\theta_1, \theta_2, \ldots$ is non-increasing, $\theta_1 = \pi_{\max}$ and $\lim_{s\to\infty} \theta_s = 0$. The proof then remains exactly the same, except that the first inequality in (27) can be now strengthened to a strict inequality. ▮

Since (27) holds for all $(i,j) \in \Phi$, both inequalities in (27) remain valid under summation over all $(i,j) \in \Phi$. Thus it follows from Lemma 9 that

$$\sum_{(i,j)\in\Phi} K(m_{i,j}(s)+1) \;\geqslant\; \sum_{(i,j)\in\Phi} \pi_{i,j} \;\geqslant\; \sum_{(i,j)\in\Phi} K m_{i,j}(s) \tag{28}$$

These inequalities lead to upper and lower bounds on the constant $K = K(s)$ in Lemma 9. Since $\sum_{(i,j)\in\Phi} m_{i,j}(s) = s$ while $\sum_{(i,j)\in\Phi} \pi_{i,j} = n$, we conclude from (28) that

$$\frac{n}{s} \;\geqslant\; K(s) \;\geqslant\; \frac{n}{s+qn} \tag{29}$$

Next, we define the ***normalized multiplicity matrix*** $\mathcal{M}'(\Pi,s) = [\mu_{i,j}(s)]$ and the ***normalized reliability matrix*** $\Pi' = [\pi'_{i,j}]$ as follows: $\mu_{i,j}(s) = m_{i,j}(s)/s$ and $\pi'_{i,j} = \pi_{i,j}/n$ for all $(i,j) \in \Phi$. It is clear from these definitions that $\langle \mathcal{M}', \mathbf{1}\rangle = \langle \Pi', \mathbf{1}\rangle = 1$, where $\mathbf{1}$ denotes the all-one matrix. The following theorem is the key result of this subsection: the theorem shows that the optimal multiplicity matrix $\mathcal{M}(\Pi,s)$ becomes proportional to Π as $s \to \infty$.

Theorem 10. *As $s \to \infty$, the normalized multiplicity matrix converges to the normalized reliability matrix: $\mathcal{M}'(\Pi,s) \longrightarrow_{s\to\infty} \Pi'$. In other words, for every $\varepsilon > 0$, there exists an s_0 such that for all $s \geqslant s_0$ we have*

$$\left|\pi'_{i,j} - \mu_{i,j}(s)\right| \;=\; \left|\frac{\pi_{i,j}}{n} - \frac{m_{i,j}(s)}{s}\right| \;\leqslant\; \varepsilon \qquad \textit{for all } (i,j) \in \Phi \tag{30}$$

Proof. It follows from Lemma 9 that for all s, there exists a constant $K(s)$ such that $1 \geqslant \pi_{i,j}/K(s) - m_{i,j}(s) \geqslant 0$ for all $(i,j) \in \Phi$. Dividing this inequality by s, we obtain

$$\frac{1}{s} \;\geqslant\; \frac{\pi_{i,j}}{sK(s)} - \mu_{i,j}(s) \;\geqslant\; 0 \tag{31}$$

From the bounds on $K(s)$ in (29), we conclude that $\pi'_{i,j} \leqslant \pi_{i,j}/sK(s) \leqslant \pi'_{i,j} + q\pi_{i,j}/s$. Combining this with (31), we get

$$\frac{1}{s} \;\geqslant\; \pi'_{i,j} - \mu_{i,j}(s) \;\geqslant\; -\frac{q\pi_{\max}}{s} \tag{32}$$

It follows that for all $s \geqslant \max\{1/\varepsilon, \pi_{\max}q/\varepsilon\} = \pi_{\max}q/\varepsilon$, the bound in (30) holds for all $(i,j) \in \Phi$. Thus $s_0 = \lceil \pi_{\max}q/\varepsilon \rceil$. ▮

Asymptotically, for a large number s of interpolation points — and, hence, for a large cost — a constraint on the cost $\mathcal{C}(M) = {}^1\!/_2\,(\langle M,M\rangle + \langle M,\mathbf{1}\rangle)$ is equivalent to a constraint on the L_2-norm $\sqrt{\langle M,M\rangle}$ of the multiplicity matrix. It is obvious that for a fixed norm $\sqrt{\langle M,M\rangle}$, maximizing the expected score $\langle M,\Pi\rangle$ is equivalent to maximizing the correlation between M and Π, which is clearly achieved by letting M be proportional to Π. This intuition confirms the result established in Theorem 10.

Remark. Finding the optimal multiplicity matrix $\mathcal{M}(\Pi, s)$ can be viewed as a gambling problem. Assume that a gambler has a certain wealth in the form of a maximal number of linear constraints the gambler can satisfy. The matrix Π provides all the information the gambler can use in order to place bets on interpolation points with the goal of maximizing the return, which is the score of the transmitted codeword. In this context, Theorem 10 shows that proportional betting is the asymptotically optimal gambling strategy. Proportional betting is known [5] to be the optimal strategy in the context of a *fair horse race*. However, these results do not appear to be related to Theorem 10 in an obvious way.

We conclude this section with a geometric characterization of the (asymptotic) decoding regions of our soft-decision decoding algorithm. To start with, the following simple lemma essentially recasts Theorem 3 in slightly different terms.

Lemma 11. *For a given multiplicity matrix M, the algebraic soft-decision decoding algorithm outputs a list that contains a codeword $\underline{c} \in \mathbb{C}_q(n,k)$ if*

$$\frac{\langle M, [\underline{c}]\rangle}{\sqrt{\langle M, M\rangle + \langle M, \mathbf{1}\rangle}} \geqslant \sqrt{k-1} \tag{33}$$

Proof. The lemma follows from Corollary 5 by observing that $\mathcal{S}_M(\underline{c}) = \langle M, [\underline{c}]\rangle$ and $2\mathcal{C}(M) = \langle M, M\rangle + \langle M, \mathbf{1}\rangle$ by definition. ∎

Theorem 10 and Lemma 11 lead to a precise characterization of the performance limits of our algorithm as the number of interpolation points approaches infinity. In the following theorem and its corollaries, $o(1)$ denotes a function of s that tends to zero as $s \to \infty$.

Theorem 12. *The algebraic soft-decision decoding algorithm outputs a list that contains a codeword $\underline{c} \in \mathbb{C}_q(n,k)$ if*

$$\frac{\langle \Pi, [\underline{c}]\rangle}{\sqrt{\langle \Pi, \Pi\rangle}} \geqslant \sqrt{k-1} + o(1) \tag{34}$$

Proof. Substituting the optimal multiplicity matrix $\mathcal{M}(\Pi, s)$ in (33) and normalizing (dividing by s the numerator and the denominator), we obtain the equivalent condition

$$\frac{\langle \mathcal{M}'(\Pi, s), [\underline{c}]\rangle}{\sqrt{\langle \mathcal{M}'(\Pi, s), \mathcal{M}'(\Pi, s)\rangle + \frac{1}{s}}} \geqslant \sqrt{k-1} \tag{35}$$

It follows from Theorem 10 that for $s \to \infty$, one can replace $\mathcal{M}'(\Pi, s)$ in (35) by Π', which upon re-normalization yields (34). More explicitly, we have

$$\frac{\langle \mathcal{M}'(\Pi, s), [\underline{c}]\rangle}{\sqrt{\langle \mathcal{M}'(\Pi, s), \mathcal{M}'(\Pi, s)\rangle + \frac{1}{s}}} \geqslant \frac{\langle \Pi, [\underline{c}]\rangle - \frac{n^2}{s}}{\sqrt{\langle \Pi, \Pi\rangle + \frac{(2q\pi_{\max}+1)n^2}{s} + \frac{q^3 n^3 \pi_{\max}^2}{s^2}}} = \frac{\langle \Pi, [\underline{c}]\rangle}{\sqrt{\langle \Pi, \Pi\rangle}} + o(1)$$

where the first inequality follows from (32) after some straightforward manipulations. In conjunction with (35), this completes the proof. ∎

Finally, Theorem 12 has a particularly nice interpretation if the reliability matrix Π and the codeword $[\underline{c}]$ are viewed as vectors in the qn-dimensional Euclidean space $\mathbb{R}^{qn}$.

Corollary 13. *Let $\beta = \beta(\Pi, \underline{c})$ denote the angle in $\mathbb{R}^{qn}$ between Π and $[\underline{c}]$. Then the algebraic soft-decision decoding algorithm outputs a list that contains $\underline{c}$ if*

$$\cos\beta \;\geqslant\; \sqrt{R} \;+\; o(1)$$

Proof. Follows from Theorem 12 and the identity $\langle \Pi, [\underline{c}] \rangle = \sqrt{n\,\langle \Pi, \Pi \rangle}\cos\beta$. ∎

Thus the asymptotic decoding regions of our algorithm are *spherical cones* in the Euclidean space $\mathbb{R}^{qn}$, extending from the origin to the surface of a sphere $\mathfrak{S}$ of radius $\sqrt{n}$. The codeword $[\underline{c}]$ is a point of $\mathfrak{S}$, and the line connecting the origin to this point constitutes the central axis of the spherical cone. The angle of each spherical cone is $\cos^{-1}\sqrt{R}$. Notice that since the algorithm is a list-decoding algorithm, its decoding regions are not disjoint: the spherical cones of angle $\cos^{-1}\sqrt{R}$ are overlapping. Also notice that we are concerned only with the positive 2^{qn}-part of the Euclidean space $\mathbb{R}^{qn}$ (which consists of points with all coordinates nonengative), since all the entries of both Π and $[\underline{c}]$ are nonnegative.

It follows from Theorem 2 that the asymptotic (for $m \to \infty$) decoding regions of the Guruswami-Sudan [11] algorithm are *spherical caps* on the surface of $\mathfrak{S}$ of the same spherical angle $\cos^{-1}\sqrt{R}$, but the decoding process involves projecting Π onto a point $[\underline{y}]$ on the surface of $\mathfrak{S}$ in a nonlinear fashion, according to equation (9). Finally, the decoding regions of conventional Berlekamp-Welch [28] hard-decision decoding are also spherical caps on the surface of $\mathfrak{S}$ and the same nonlinear projection is employed, but the spherical angle of these caps is only $\cos^{-1}(1/2 + 1/2R)$, and they are non-overlapping.

5.2. Asymptotic analysis for long codes

As noted in Section 4, from the point of view of the receiver, the transmitted codeword is a random vector $\boldsymbol{\mathcal{X}} = (\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n)$ whose *a posteriori* probability distribution is given by (18). For notational convenience, let us introduce two random variables:

$$\mathcal{Z} \overset{\text{def}}{=} \langle M, [\boldsymbol{\mathcal{X}}] \rangle \qquad \text{and} \qquad \mathcal{Z}^* \overset{\text{def}}{=} \frac{\langle M, [\boldsymbol{\mathcal{X}}] \rangle}{\sqrt{n\,\langle M, M \rangle + n\,\langle M, \mathbf{1} \rangle}}$$

The key result of this subsection is the following theorem which shows that as $n \to \infty$, the random variable $\mathcal{Z}^*$ converges to its expected value.

Theorem 14. *Suppose that a $q \times n$ reliability matrix Π is given, and let M be an arbitrary $q \times n$ multiplicity matrix. Then for any $\varepsilon > 0$, we have*

$$\Pr\Big\{ \big|\mathcal{Z}^* - \mathsf{E}\{\mathcal{Z}^*\}\big| \geqslant \varepsilon \Big\} \;\leqslant\; \frac{1}{n\varepsilon^2} \tag{36}$$

Proof. Consider the random variable $\mathcal{Z} = \langle M, [\boldsymbol{\mathcal{X}}] \rangle = M(\mathcal{X}_1, 1) + \cdots + M(\mathcal{X}_n, n)$ and define $\mathcal{Z}_j = M(\mathcal{X}_j, j)$ for $j = 1, 2, \ldots, n$. Thus $\mathcal{Z}_j$ is the entry found in the j-th column of M in the row indexed by $\mathcal{X}_j$. The distribution of $\mathcal{X}_j$, computed by marginalizing the distribution of $\boldsymbol{\mathcal{X}}$ in (18), is given by

$$\Pr\Big(\mathcal{X}_j = \alpha_i\Big) \;=\; \pi_{i,j} \qquad \text{for} \;\; i = 1,2,\ldots,q \;\; \text{and} \;\; j = 1,2,\ldots,n$$

Using this distribution, we find that $\mathsf{E}\{\mathcal{Z}_j\} = \sum_{i=1}^{q} m_{i,j}\pi_{i,j}$ and $\mathsf{E}\{\mathcal{Z}_j^2\} = \sum_{i=1}^{q} m_{i,j}^2\pi_{i,j}$. The key observation is this: since the random variables $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n$ are independent, so are $\mathcal{Z}_1, \mathcal{Z}_2, \ldots, \mathcal{Z}_n$. Hence

$$\mathsf{Var}(\mathcal{Z}) = \sum_{j=1}^{n} \mathsf{Var}(\mathcal{Z}_j) \;=\; \sum_{j=1}^{n}\left(\sum_{i=1}^{q} m_{i,j}^2\pi_{i,j} - \left(\sum_{i=1}^{q} m_{i,j}\pi_{i,j}\right)^{2}\right) \;\leqslant\; \sum_{j=1}^{n}\sum_{i=1}^{q} m_{i,j}^2 \;=\; \langle M, M\rangle$$

The theorem now follows by a straightforward application of the Chebycheff inequality [20, p. 193] to the random variable $\mathcal{Z}^*$. Thus

$$\Pr\Big\{\, \big|\mathcal{Z}^* - \mathsf{E}\{\mathcal{Z}^*\}\big| \geqslant \varepsilon \Big\} \;\leqslant\; \frac{\mathsf{Var}(\mathcal{Z}^*)}{\varepsilon^2} \;=\; \frac{\mathsf{Var}(\mathcal{Z})}{\varepsilon^2\Big(n\,\langle M, M\rangle + n\,\langle M, \mathbf{1}\rangle\Big)} \;\leqslant\; \frac{1}{n\varepsilon^2} \qquad \blacksquare$$

Remark. The proof of Theorem 14 is essentially similar to a well-known proof of the weak law of large numbers. We note that using the strong law of large numbers, it is possible to show that as $n \to \infty$, the random variable $\mathcal{Z}^*$ equals its expectation with probability 1.

We can use Theorem 14 to derive a relationship between the probability P_e of list-decoding failure, the expected score, and the rate R of the Reed-Solomon code. Indeed, we have

$$P_e \;\leqslant\; \Pr\Big\{\mathcal{Z} \leqslant \Delta_{1,k-1}(\mathcal{C})\Big\} \;\leqslant\; \Pr\Big\{\mathcal{Z} \leqslant \sqrt{2k\mathcal{C}}\Big\} \;=\; \Pr\Big\{\mathcal{Z}^* \leqslant \sqrt{R}\Big\} \tag{37}$$

where the first inequality follows from Theorem 3, and the second from the fact that $\Delta_{1,k-1}(\mathcal{C}) < \sqrt{2k\mathcal{C}}$. In view of Theorem 14, this immediately implies (16). Namely,

$$\sqrt{R} \;\leqslant\; \mathsf{E}\{\mathcal{Z}^*\} - \frac{1}{\sqrt{\varepsilon n}} \quad\Rightarrow\quad P_e \leqslant \varepsilon$$

We can also derive a bound in the opposite direction, but to do so we need two assumptions. First, we assume that the first inequality in (37) holds with equality. This is tantamount to assuming that condition (15) of Theorem 3 is not only sufficient but also necessary for successful list decoding. Strictly speaking, this is not true. It is easy to construct examples where $\mathcal{Q}_M(X,Y)$ has a factor $Y - f(X)$, with $f(X)$ evaluating to a codeword $\underline{c} \in \mathbb{C}_q(n,k)$, and yet $\langle M, [\underline{c}]\rangle \leqslant \Delta_{1,k-1}(\mathcal{C})$. In fact, such situations do arise in simulations. However, they occur so infrequently that this phenomenon has no effect on the overall performance. To be specific, the approximation $P_e \simeq \Pr\{\mathcal{Z} \leqslant \Delta_{1,k-1}(\mathcal{C})\}$ is usually accurate up to the second significant digit. The second assumption is that $\Delta_{1,k-1}(\mathcal{C})$ is well approximated by $\sqrt{2(k-1)\mathcal{C}}$. This is certainly true for large costs, as can be seen from Figure 2. Combining the two assumptions with Theorem 14 produces the bound in (17).

6. Performance analysis for a fixed list size

In this section, we study the performance achieved by our soft-decoding algorithm under a constraint which guarantees that the number of codewords on the list produced by the decoder does not exceed a given bound L. The key analytical result in this section is Theorem 17. This theorem extends Theorem 12 by providing a bound on how quickly the decoding algorithm converges to the asymptotic performance as a function of L. The analytical results are confirmed by simulations for both high-rate and low-rate codes.

We start with two lemmas. As observed in Section 4, the number of codewords on the list produced by the soft-decision decoder is upper-bounded by $\deg_{0,1} \mathcal{Q}_M(X,Y)$, where $\mathcal{Q}_M(X,Y)$ is the interpolation polynomial. This leads to the following lemma.

Lemma 15. *The number of codewords on the list produced by the soft-decision decoder for a given multiplicity matrix M does not exceed*

$$\mathcal{L}_k(M) \;\overset{\text{def}}{=}\; \frac{\sqrt{\langle M,M\rangle + \langle M,\mathbf{1}\rangle}}{\sqrt{k-1}} \tag{38}$$

Proof. The size of the list is at most $\deg_{0,1} \mathcal{Q}_M(X,Y)$. By the definition of weighted-degree, we have $\deg_{0,1} \mathcal{Q}_M(X,Y) \leqslant \deg_{1,k-1} \mathcal{Q}_M(X,Y)/(k-1)$. Now

$$\frac{\deg_{1,k-1} \mathcal{Q}_M(X,Y)}{k-1} \;\leqslant\; \frac{\Delta_{1,k-1}(\mathcal{C})}{k-1} \;\leqslant\; \frac{\sqrt{\langle M,M\rangle + \langle M,\mathbf{1}\rangle}}{\sqrt{k-1}}$$

where the first inequality follows from (13) and (14), while the second inequality follows from the definition of the cost $\mathcal{C} = \mathcal{C}(M)$, Lemma 1, and (14). ∎

Let Π be a given reliability matrix, and let $\mathcal{M}(\Pi,s)$ be the corresponding multiplicity matrix produced by Algorithm A. For convenience, we define $\mathcal{M}(\Pi,0)$ as the all-zero $q\times n$ matrix. Let $\mathcal{J}$ denote a $q\times n$ matrix all of whose entries are nonnegative real numbers not exceeding 1. We write $\mathcal{J}^*$ instead of $\mathcal{J}$ if all the entries in the matrix are strictly less than 1.

Lemma 16. *For every positive real number λ, there exists a nonnegative integer s, such that the matrix $\mathcal{M}(\Pi,s)$ can be written as*

$$\mathcal{M}(\Pi,s) \;=\; \lfloor \lambda\Pi \rfloor \;\overset{\text{def}}{=}\; \lambda\Pi \;-\; \mathcal{J}^* \tag{39}$$

Conversely, for every nonnegative integer s, there exists a positive real λ such that (39) *holds, possibly with $\mathcal{J}^*$ replaced by $\mathcal{J}$.*

Proof. As before, let $\pi_{\max}$ be the largest entry in Π. If $\lambda < \pi_{\max}^{-1}$, then $\mathcal{M}(\Pi,0)$ satisfies (39). Otherwise, set $K = \lambda^{-1}$, so that $0 < K \leqslant \pi_{\max}$. We know from Lemma 9 and its proof that there exists a positive integer s, such that

$$\frac{\pi_{i,j}}{K} - 1 \;<\; m_{i,j}(s) \;\leqslant\; \frac{\pi_{i,j}}{K} \qquad\qquad \text{for all } (i,j) \in \Phi \tag{40}$$

It follows from (40) that $\mathcal{M}(\Pi,s)$ is of the form (39). Conversely, given $\mathcal{M}(\Pi,s)$, we take $\lambda = K^{-1}$, where $K = K(s) \leqslant \pi_{\max}$ is the constant derived in Lemma 9. ∎

Given $M = \mathcal{M}(\Pi, s)$, let λ be a positive real constant such that $M = \lambda\Pi - \mathcal{J}$. Such a constant exists by Lemma 16. Then

$$\langle M, M\rangle + \langle M, \mathbf{1}\rangle \;=\; \lambda^2 \langle \Pi, \Pi\rangle - \lambda \langle \Pi, 2\mathcal{J}-\mathbf{1}\rangle + \langle \mathcal{J}, \mathcal{J}-\mathbf{1}\rangle \tag{41}$$

We can use (41) and (38) to derive an expression for λ in terms of Π, $\mathcal{J}$, and $\mathcal{L}_k(M)$. Equating the right-hand side of (41) to $(k-1)\mathcal{L}_k^2(M)$, we obtain a quadratic equation in λ. Since $\langle \Pi, \Pi\rangle > 0$ and $\langle \mathcal{J}, \mathcal{J}-\mathbf{1}\rangle \leqslant 0$, this equation has one positive root and one negative root. Solving for the unique positive root yields

$$\lambda \;=\; \frac{\langle \Pi, 2\mathcal{J}-\mathbf{1}\rangle}{2\,\langle \Pi, \Pi\rangle} + \sqrt{\frac{\langle \Pi, 2\mathcal{J}-\mathbf{1}\rangle^2}{4\,\langle \Pi, \Pi\rangle^2} + \frac{\langle \mathcal{J}, \mathbf{1}-\mathcal{J}\rangle}{\langle \Pi, \Pi\rangle} + \frac{(k-1)\mathcal{L}_k^2(M)}{\langle \Pi, \Pi\rangle}} \tag{42}$$

Suppose now that we are given a positive integer L and would like to guarantee that the number of codewords on the list produced by the soft-decision decoder does not exceeed L. In view of Lemma 15, we can do so by computing $\mathcal{L}_k(M)$ at each iteration of Algorithm A, and stopping the algorithm just before $\mathcal{L}_k(M)$ equals or exceeds $L+1$. At this point

$$L \;\leqslant\; \mathcal{L}_k(M) \;<\; L+1 \tag{43}$$

and since the number of codewords on the list produced by the decoder is an integer not exceeding $\mathcal{L}_k(M)$, this number is at most L. We will refer to this decoding procedure[*] as *algebraic soft-decoding with list-size limited to L*.

Theorem 17. *Algebraic soft-decoding with list-size limited to L produces a list that contains a codeword $\underline{c} \in \mathbb{C}_q(n,k)$ if*

$$\frac{\langle \Pi, [\underline{c}]\rangle}{\sqrt{\langle \Pi, \Pi\rangle}} \;\geqslant\; \frac{\sqrt{k-1}}{1 - \frac{1}{L}\left(\frac{1}{R^*} + \frac{\sqrt{q}}{2\sqrt{R^*}}\right)} \;=\; \sqrt{k-1}\cdot\left(1 + O\left(\tfrac{1}{L}\right)\right) \tag{44}$$

where Π is the reliability matrix derived from the channel output, $R^* = (k-1)/n$ is the rate of $\mathbb{C}_q(n,k-1)$, and the constant in $O(\cdot)$ depends only on R^* and q.

Proof. Writing $M = \lambda\Pi - \mathcal{J}$ as in Lemma 16 and using the definition of $\mathcal{L}_k(M)$ in (38), we can recast the sufficient condition (33) of Lemma 11 in the following way

$$\frac{\langle \Pi, [\underline{c}]\rangle}{\sqrt{\langle \Pi, \Pi\rangle}} \cdot \left(\lambda - \frac{\langle \mathcal{J}, [\underline{c}]\rangle}{\langle \Pi, [\underline{c}]\rangle}\right) \cdot \frac{\sqrt{\langle \Pi, \Pi\rangle}}{\mathcal{L}_k(M)\sqrt{k-1}} \;\geqslant\; \sqrt{k-1} \tag{45}$$

Using the expression for λ in (42), we now express the factor multiplying $\langle \Pi, [\underline{c}]\rangle / \sqrt{\langle \Pi, \Pi\rangle}$ on the left-hand side of (45) as $\mathcal{F}_1(\Pi, L) - \mathcal{F}_2(\Pi, L) - \mathcal{F}_3(\Pi, \underline{c}, L)$, where

$$\mathcal{F}_1(\Pi, L) \;\overset{\text{def}}{=}\; \sqrt{1 + \frac{\langle \Pi, 2\mathcal{J}-\mathbf{1}\rangle^2}{4\,\langle \Pi, \Pi\rangle\, \mathcal{L}_k^2(M)(k-1)} + \frac{\langle \mathcal{J}, \mathbf{1}-\mathcal{J}\rangle}{\mathcal{L}_k^2(M)(k-1)}} \;\geqslant\; 1 \tag{46}$$

[*]In practice, algebraic soft-decoding with list-size limited to L almost always produces lists with much less than L codewords, most often a single codeword.

$$\mathcal{F}_2(\Pi, L) \overset{\text{def}}{=} \frac{1}{\mathcal{L}_k(M)\sqrt{k-1}} \cdot \frac{\langle \Pi, \mathbf{1}-2\mathcal{J} \rangle}{2\sqrt{\langle \Pi, \Pi \rangle}} \leqslant \frac{1}{\mathcal{L}_k(M)\sqrt{k-1}} \cdot \frac{n}{2\sqrt{n/q}} \tag{47}$$

and

$$\mathcal{F}_3(\Pi, \underline{c}, L) \overset{\text{def}}{=} \frac{1}{\mathcal{L}_k(M)\sqrt{k-1}} \cdot \frac{\langle \mathcal{J}, [\underline{c}] \rangle \sqrt{\langle \Pi, \Pi \rangle}}{\langle \Pi, [\underline{c}] \rangle} \leqslant \frac{1}{\mathcal{L}_k(M)\sqrt{k-1}} \cdot \frac{n}{\sqrt{k-1}} \tag{48}$$

To obtain the inequality in (47), we have used the fact that $\langle \Pi, \mathbf{1}-2\mathcal{J} \rangle \leqslant \langle \Pi, \mathbf{1} \rangle = n$ and $\langle \Pi, \Pi \rangle \geqslant n/q$. To obtain the inequality in (48), we have made use of the following two observations. First, we have $\langle \mathcal{J}, [\underline{c}] \rangle \leqslant \langle \mathbf{1}, [\underline{c}] \rangle = n$. Secondly, if Π and $\underline{c}$ are such that (44) holds, then a fortiori $\langle \Pi, [\underline{c}] \rangle / \sqrt{\langle \Pi, \Pi \rangle} \geqslant \sqrt{k-1}$. Since $L \leqslant \mathcal{L}_k(M)$ by (43), it follows from (47) and (48), respectively, that $\mathcal{F}_2(\Pi, L) \leqslant \sqrt{q}/2L\sqrt{R^*}$ and $\mathcal{F}_3(\Pi, \underline{c}, L) \leqslant 1/LR^*$. In conjunction with (46) and (45), this completes the proof of the theorem. ∎

We observe that Theorem 17 is a very loose bound. The actual performance of algebraic soft-decoding with list-size limited to L is usually orders of magnitude better than that predicted by (44). In the proof of Theorem 17, we have used the inequality $\langle \Pi, \Pi \rangle \geqslant n/q$, which is a weak lower bound since $\langle \Pi, \Pi \rangle \simeq n$ for SNRs of practical interest. Replacing the bound n/q on the right-hand side of (47) by the actual value $\langle \Pi, \Pi \rangle$, we thus obtain a somewhat stronger bound, which guarantees that $\underline{c} \in \mathbb{C}_q(n,k)$ is on the list produced by the soft-decision decoder, provided

$$\frac{\langle \Pi, [\underline{c}] \rangle}{\sqrt{\langle \Pi, \Pi \rangle}} \geqslant \frac{\sqrt{k-1}}{1 - \frac{1}{L}\left(\frac{1}{R^*} + \frac{1}{2\sqrt{R^*}} \cdot \frac{\sqrt{n}}{\sqrt{\langle \Pi, \Pi \rangle}}\right)} \simeq \frac{\sqrt{k-1}}{1 - \frac{1}{L}\left(\frac{1}{R^*} + \frac{1}{2\sqrt{R^*}}\right)} \tag{49}$$

This works well for large L, although (49) is still a loose bound for moderate list sizes. Nevertheless, the significance of Theorem 17 is that it proves convergence to the asymptotic performance *at least* as fast as $O(1/L)$. Furthermore, the theorem shows that the size of the list required to approach the asymptotic performance within any given constant does not depend on the length of the code. Note that for Reed-Solomon codes, the right-hand side of (44) depends on the length n indirectly via the expression $\sqrt{q}/2\sqrt{R^*}$. However, for algebraic-geometric codes, we can have arbitrary lengths for a fixed q. If one is willing to accept the approximation on the right-hand side of (49), then the size of the list depends *only* on the rate R^*, for both Reed-Solomon and algebraic-geometric codes.

In addition to the analysis of Theorem 17, we have performed extensive simulations of algebraic soft-decoding with list-size limited to L for various Reed-Solomon codes over GF(256). As the running channel model, we have assumed an AWGN channel with a 256-QAM signal constellation. The 256 constellation points were matched to the 256 elements of GF(256) in an arbitrary manner. The reliability matrix Π was computed by measuring the distance from the channel output to the four nearest constellation points. All the entries in Π were normalized and quantized to 8 bits of precision.

Simulation results for the $(255, 144, 112)$ Reed-Solomon code of rate ~0.56 are summarized in Figure 3. One can see from Figure 3 that at codeword error-rates of 10^{-5} and lower, algebraic soft-decision decoding provides a coding gain of about 1.5 dB, whereas GMD de-

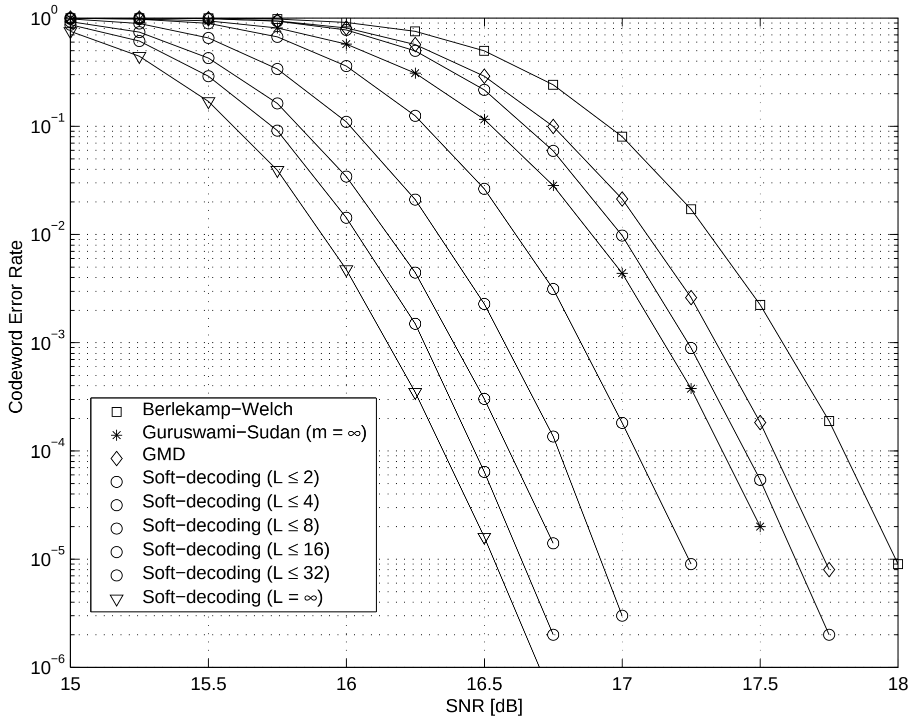

Figure 3. *Performance of algebraic soft-decision decoding for the* $(255, 144, 112)$ *Reed-Solomon code with* 256-QAM *modulation on an AWGN channel*

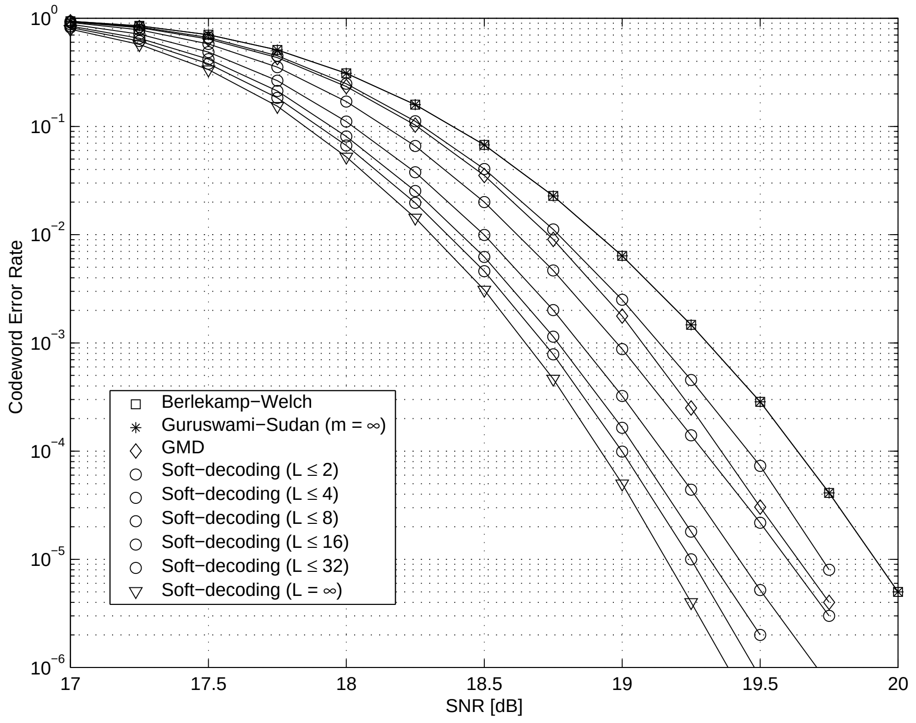

Figure 4. *Performance of algebraic soft-decoding for the* $(204, 188, 17)$ *shortened Reed-Solomon code with* 256-QAM *modulation on an AWGN channel*

coding and Guruswami-Sudan decoding achieve coding gains of about 0.2 dB and 0.4 dB, respectively, compared to conventional hard-decision decoding. Although the 1.5 dB coding gain corresponds to asymptotic performance (cf. Theorem 12), it is evident from Figure 3 that most of this gain can be obtained with very small list sizes. A list of size $L = 4$ already outperforms both GMD and Guruswami-Sudan decoding by a substantial margin, while a list of size $L = 32$ approaches the asymptotic performance to within 0.1 dB.

Simulation results for the $(204, 188, 17)$ shortened Reed-Solomon code of rate ~0.92 are presented in Figure 4. We observe that this code, in conjunction with a 256-QAM signal constellation, is implemented today in certain satellite communications systems. Here, algebraic soft-decision decoding provides an ultimate coding gain of about 0.75 dB. The fact that the asymptotic coding gain decreases with the rate of a code is to be expected since list-decoding, in general, is less effective for high-rate codes. In fact, the asymptotic performance of Guruswami-Sudan list decoding coincides with that of the conventional Berlekamp-Wech decoding for the $(204, 188, 17)$ code: the Guruswami-Sudan decoder finds all codewords within Hamming distance of $\lfloor 204(1 - \sqrt{0.92}) \rfloor = \lfloor 8.16 \rfloor = (17-1)/2$ from the (hard-decision) channel output (cf. Theorem 2). In contrast, soft-decision list decoding does provide a significant coding gain. As in the case of half-rate codes, most of this gain can be achieved with small list sizes. Moreover, one can see from Figure 4 that the coding gain grows with SNR. Extrapolating the simulation results to error rates of about 10^{-10} (that are of interest for many applications), one should expect coding gains in excess of about 1.0 dB for high-rate as well as low-rate Reed-Solomon codes.

7. Conclusions

We have shown that interpolation-based decoding can be used to devise an efficient soft-decision decoding algorithm for Reed-Solomon codes. The soft-decoding algorithm outperforms both GMD decoding and Guruswami-Sudan list-decoding by a substantial margin.

The focus of this paper has been the performance achievable in a probabilistic setting, where the channel output is characterized in terms of a posteriori probabilities rather than error patterns. This is quite different from several recent papers [12, 17] which focus on a combinatorial setting, and provide guarantees on the number (and type) of errors that can be corrected on certain hard-decision channels. In particular, for long codes, the criterion derived herein for the computation of a multiplicity matrix allows for reliable transmission at the highest possible rate, although this is not necessarily the criterion that maximizes the number of correctable errors.

The asymptotic performance of the proposed soft-decoding algorithm for a large number of interpolation points or, equivalently, for large lists has been characterized in terms of simple geometric conditions. Moreover, it has been shown that that the asymptotic performance can be approached arbitrarily closely with list sizes that are bounded by a constant, even as the length of a code grows beyond all bounds.

Appendix A. On the underlying probabilistic model

In Section 4, in order to convert posterior probabilities (the reliability matrix Π) into interpolation points (the multiplicity matrix M), we regard the transmitted codeword as a random vector $\boldsymbol{\mathcal{X}} = (\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n) \in \mathscr{X}^n$ and use the following probability distribution

$$P(x_1, x_2, \ldots, x_n) \;=\; \prod_{j=1}^{n} \Pr\Big(\mathcal{X}_j = x_j \mid \mathcal{Y}_j = y_j\Big) \;=\; \prod_{j=1}^{n} \Pi(x_j, j)$$

where $\underline{y} = (y_1, y_2, \ldots, y_n) \in \mathscr{Y}^n$ is the vector observed at the channel output (cf. equation (18) of Section 4). Recall that this distribution corresponds to the following scenario: a vector $\boldsymbol{\mathcal{X}}$ is drawn uniformly at random from the space $\mathbb{F}_q^n$ and transmitted over a memoryless channel characterized by (6); thereupon the vector $\underline{y} \in \mathscr{Y}^n$ is observed at the channel output. Up to certain natural assumptions, this is indeed what happens, *except* that the transmitted codeword $\boldsymbol{\mathcal{X}}$ is drawn uniformly at random from the code $\mathbb{C}_q(n,k)$ rather than the entire space $\mathbb{F}_q^n$. Thus the *a priori* distribution of $\boldsymbol{\mathcal{X}}$ is $\Pr(\boldsymbol{\mathcal{X}} = \underline{x}) = \mathcal{I}_{\mathbb{C}}(\underline{x})/q^k$, where $\mathcal{I}_{\mathbb{C}}(\underline{x}) : \mathbb{F}_q^n \to \{0,1\}$ is the indicator function for $\mathbb{C}_q(n,k)$ defined by

$$\mathcal{I}_{\mathbb{C}}(\underline{x}) \;\overset{\text{def}}{=}\; \begin{cases} 1 & \text{if } \underline{x} \in \mathbb{C}_q(n,k) \\ 0 & \text{otherwise} \end{cases}$$

Given the channel observations $\underline{y} = (y_1, y_2, \ldots, y_n) \in \mathscr{Y}^n$ one can easily compute the true posterior probability distribution of $\boldsymbol{\mathcal{X}}$ as follows

$$P^*(x_1, x_2, \ldots, x_n) \;\overset{\text{def}}{=}\; \Pr\Big(\boldsymbol{\mathcal{X}} = \underline{x} \mid \boldsymbol{\mathcal{Y}} = \underline{y}\Big) \;=\; \gamma\, \mathcal{I}_{\mathbb{C}}(\underline{x}) \prod_{j=1}^{n} \Pi(x_j, j) \qquad (50)$$

The normalization constant γ in (50) is given by

$$\gamma \;\overset{\text{def}}{=}\; \frac{1}{\displaystyle\sum_{\underline{x} \in \mathbb{C}} \prod_{j=1}^{n} \Pi(x_j, j)} \;=\; q^{n-k}\, \frac{\prod_{j=1}^{n} f_{\mathcal{Y}_j}(y_j)}{f_{\boldsymbol{\mathcal{Y}}}(\underline{y})} \qquad (51)$$

where $f_{\boldsymbol{\mathcal{Y}}}(\cdot)$ is the probability-density function of the channel output $\boldsymbol{\mathcal{Y}} = (\mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_n)$ (we assume w.l.o.g. that $\boldsymbol{\mathcal{Y}}$ is continuous), and $f_{\mathcal{Y}_j}(\cdot)$ are the marginal probability densities derived from $f_{\boldsymbol{\mathcal{Y}}}(\cdot)$. The expression in (50) follows by repeated application of the Bayes rule, first to $\Pr(\boldsymbol{\mathcal{X}} = \underline{x} \,|\, \boldsymbol{\mathcal{Y}} = \underline{y})$ and then to $\Pr(\mathcal{Y}_j = y_j \mid \mathcal{X}_j = x_j)$. Hence the precise optimization problem we would like to solve is

$$M_{\text{opt}}(\Pi, \mathcal{C}) \;\overset{\text{def}}{=}\; \text{argmax}_{M \in \mathscr{M}(\mathcal{C})} \mathsf{E}_{P^*}\{\mathcal{S}_M(\boldsymbol{\mathcal{X}})\} \qquad (52)$$

where, in contrast to (19), the expectation $\mathsf{E}_{P^*}\{\cdot\}$ is taken with respect to the true posterior distribution (50). While (52) gives a natural optimality criterion for the computation of the multiplicity matrix, we shall see that the computation itself is likely to be intractable.

There are two sources of difficulty in performing the maximization in (52). One of these has to do with the fact that computing $P^*(\underline{x})$ is difficult, even for a single input vector

$\underline{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{F}_q^n$. While $\mathcal{I}_{\mathbb{C}}(\underline{x})$ and $\prod_{j=1}^n \Pi(x_j, j)$ are easy to evaluate, it can be shown that computing γ in (51) for an arbitrary reliability matrix Π and an arbitrary linear code $\mathbb{C}$ is NP-hard. This difficulty, however, can be avoided as follows. Let

$$\Psi(\underline{x}) \overset{\text{def}}{=} \frac{P^*(\underline{x})}{\gamma} = \mathcal{I}_{\mathbb{C}}(\underline{x}) \prod_{j=1}^{n} \Pi(x_j, j) \tag{53}$$

be a density function. Given a multiplicity matrix M, let us formally define the expected score with respect to $\Psi(\cdot)$ as follows

$$\mathsf{E}_{\Psi}\{\mathcal{S}_M(\boldsymbol{\mathcal{X}})\} \overset{\text{def}}{=} \sum_{\underline{x} \in \mathscr{X}^n} \mathcal{S}_M(\underline{x}) \Psi(\underline{x}) = \sum_{\underline{x} \in \mathbb{F}_q^n} \sum_{j=1}^{n} M(x_j, j) \Psi(\underline{x}) \tag{54}$$

Then it is easy to see from (53) and (54) that $\mathsf{E}_{P^*}\{\mathcal{S}_M(\boldsymbol{\mathcal{X}})\}$ and $\mathsf{E}_{\Psi}\{\mathcal{S}_M(\boldsymbol{\mathcal{X}})\}$ differ by a factor of γ that does not depend on the multiplicity matrix M. Thus the knowledge of γ is not essential for the computation of argmax in (52), and we have

$$M_{\text{opt}}(\Pi, \mathcal{C}) \overset{\text{def}}{=} \text{argmax}_{M \in \mathscr{M}(\mathcal{C})} \mathsf{E}_{P^*}\{\mathcal{S}_M(\boldsymbol{\mathcal{X}})\} = \text{argmax}_{M \in \mathscr{M}(\mathcal{C})} \mathsf{E}_{\Psi}\{\mathcal{S}_M(\boldsymbol{\mathcal{X}})\} \tag{55}$$

Unfortunately, the second difficulty in the optimization of (52) and (55) is inherent in the presence of the indicator function $\mathcal{I}_{\mathbb{C}}(\cdot)$ in both $P^*(\cdot)$ and $\Psi(\cdot)$. Specifically, we now show that given a polynomial-time algorithm for the computation of $M_{\text{opt}}(\Pi, \mathcal{C})$ in (55), one could devise a polynomial-time algorithm for maximum-likelihood hard-decision decoding of $\mathbb{C}_q(n,k)$. If $\mathbb{C}_q(n,k)$ is a general linear code, the latter task is known [3] to be NP-hard.

More precisely, let q be a fixed prime power and let $d(\cdot,\cdot)$ denote the Hamming distance; then, the following decision problem

Problem: MAXIMUM-LIKELIHOOD DECODING
Instance: Positive integers n, k, t, an $(n-k) \times n$ matrix H over $\mathbb{F}_q$, and a vector $\underline{y} \in \mathbb{F}_q^n$.
Question: Is there a vector $\underline{c} \in \mathbb{F}_q^n$ such that $d(\underline{c}, \underline{y}) \leqslant t$ and $H\underline{c}^T = \mathbf{0}$?

was shown to be NP-complete by Berlekamp, McEliece, and van Tilborg [3]. Let $\mathbb{Q}$ denote the field of rational numbers. In this appendix, we exhibit a polynomial transformation from MAXIMUM-LIKELIHOOD DECODING to the following decision problem

Problem: OPTIMAL MULTIPLICITY MATRIX
Instance: Positive integers n, k, and $\mathcal{C}$, an $(n-k) \times n$ matrix H over $\mathbb{F}_q$ which defines a code $\mathbb{C}_q(n,k)$, a $q \times n$ reliability matrix Π over $\mathbb{Q}$, and a rational number β.
Question: Is there a matrix $M \in \mathscr{M}(\mathcal{C})$ such that $\mathsf{E}_{\Psi}\{\mathcal{S}_M(\boldsymbol{\mathcal{X}})\} \geqslant \beta$?

It is easy to see that OPTIMAL MULTIPLICITY MATRIX is just a re-formulation of the optimization problem (55) as a decision problem. Notice that this decision problem is not necessarily in NP, since given a putative solution $M \in \mathscr{M}(\mathcal{C})$, there is no obvious way to verify that $\mathsf{E}_{\Psi}\{\mathcal{S}_M(\boldsymbol{\mathcal{X}})\} \geqslant \beta$ in polynomial time.

Theorem 18. Optimal Multiplicity Matrix *is NP-hard.*

Proof. We reduce from Maximum-Likelihood Decoding. Given an instance $\{H, \underline{y}, t\}$ of Maximum-Likelihood Decoding, we generate an instance of Optimal Multiplicity Matrix as follows. Fix a rational number ϵ such that $1 > \epsilon > q^k/(q^k + q - 1)$ and let $\delta = (1-\epsilon)/(q-1)$. This choice of ϵ and δ ensures that $\epsilon + (q-1)\delta = 1$ and $\epsilon/\delta > q^k$. In terms of ϵ, δ, and $\underline{y}$, we set $\Pi = \epsilon\,[\underline{y}] + \delta\,(\mathbf{1} - [\underline{y}])$. The fact that $\epsilon + (q-1)\delta = 1$ implies that Π is a valid reliability matrix. We take $\beta = n\epsilon^{n-t}\delta^t$. Finally, we use the same parity-check matrix H, and set $\mathcal{C} = n$. This completes the mapping of $\{H, \underline{y}, t\}$ onto an instance $\{H, \Pi, \mathcal{C}, \beta\}$ of Optimal Multiplicity Matrix.

Suppose that $\{H, \underline{y}, t\}$ is a "YES" instance of Maximum-Likelihood Decoding. Then there exists a codeword $\underline{c} \in \mathbb{C}_q(n,k)$ such that $d(\underline{c}, \underline{y}) \leqslant t$. Let $M = [\underline{c}]$. It is easy to see that $\mathcal{C}(M) = n$, and so $M \in \mathscr{M}(\mathcal{C})$ for $\mathcal{C} = n$. Furthermore

$$\mathsf{E}_\Psi\{\mathcal{S}_M(\boldsymbol{\mathcal{X}})\} \;=\; \sum_{\underline{x} \in \mathscr{X}^n} \langle M, [\underline{x}] \rangle\, \Psi(\underline{x}) \;=\; \sum_{\underline{x} \in \mathbb{C}_q(n,k)} \langle [\underline{c}], [\underline{x}] \rangle \prod_{j=1}^{n} \Pi(x_j, j) \;\geqslant\; n \prod_{j=1}^{n} \Pi(c_j, j)$$

where the inequality follows by retaining a single term in the summation over $\underline{x} \in \mathbb{C}_q(n,k)$ that corresponds to $\underline{x} = \underline{c}$. With the reliability matrix given by $\Pi = \epsilon\,[\underline{y}] + \delta\,(\mathbf{1} - [\underline{y}])$, we further conclude that

$$\mathsf{E}_\Psi\{\mathcal{S}_M(\boldsymbol{\mathcal{X}})\} \;\geqslant\; n \prod_{j=1}^{n} \Pi(c_j, j) \;=\; n\,\epsilon^{n-d(\underline{c},\underline{y})}\,\delta^{d(\underline{c},\underline{y})} \;\geqslant\; n\,\epsilon^{n-t}\delta^t \;=\; \beta$$

where the last inequality follows from the fact that $d(\underline{c}, \underline{y}) \leqslant t$ and $\delta \leqslant \epsilon$. Therefore if $\{H, \underline{y}, t\}$ is a "YES" instance of Maximum-Likelihood Decoding then $\{H, \Pi, \mathcal{C}, \beta\}$ is also a "YES" instance of Optimal Multiplicity Matrix.

Now suppose that $\{H, \underline{y}, t\}$ is a "NO" instance of Maximum-Likelihood Decoding. Then $d(\underline{x}, \underline{y}) \geqslant t+1$ for all $\underline{x} \in \mathbb{C}_q(n,k)$. Observe that for any matrix $M \in \mathscr{M}(n)$ and any vector $\underline{x} \in \mathbb{F}_q^n$, we have $\langle M, [\underline{x}] \rangle \leqslant \langle M, \mathbf{1} \rangle \leqslant \mathcal{C}(M) = n$. It follows that

$$\mathsf{E}_\Psi\{\mathcal{S}_M(\boldsymbol{\mathcal{X}})\} \;=\; \sum_{\underline{x} \in \mathbb{C}_q(n,k)} \langle M, [\underline{x}] \rangle \prod_{j=1}^{n} \Pi(x_j, j) \;\leqslant\; \sum_{\underline{x} \in \mathbb{C}_q(n,k)} n\,\epsilon^{n-t-1}\delta^{t+1} \;=\; q^k \left(\frac{\delta}{\epsilon}\right) \beta \;<\; \beta$$

for any $M \in \mathscr{M}(n)$. Hence if $\{H, \underline{y}, t\}$ is a "NO" instance of Maximum-Likelihood Decoding then $\{H, \Pi, \mathcal{C}, \beta\}$ is a "NO" instance of Optimal Multiplicity Matrix. ∎

It follows from Theorem 18 that solving the optimization problem (55) for an arbitrary linear code $\mathbb{C}_q(n,k)$ and an arbitrary cost $\mathcal{C}$ is NP-hard. It is possible to argue that the original optimization problem (19) might be also NP-hard for arbitrary costs; nevertheless, Algorithm A solves this problem for certain specific costs. However, in contrast to (19), the optimization in (55) remains NP-hard even if we restrict the cost to $\mathcal{C} = n$. Furthermore, as can be seen from the proof of Theorem 18, maximizing $\mathsf{E}_{P^*}\{\mathcal{S}_M(\boldsymbol{\mathcal{X}})\}$ over all multiplicity matrices M such that $\langle M, \mathbf{1} \rangle = n$ (this is equivalent to selecting n interpolation points regardless of the cost) is still NP-hard. The analogous problem for $\mathsf{E}_P\{\mathcal{S}_M(\boldsymbol{\mathcal{X}})\}$, where $P(\cdot)$ is the distribution in (18) is trivial: it is solved by allocating all the n points at the position of the largest entry in Π.

Finally, one might argue that while the Optimal Multiplicity Matrix problem has to do with arbitrary linear codes over $\mathbb{F}_q$, the codes involved in the optimization task (55) are Reed-Solomon codes and thus have a lot of structure. In this context, Theorem 18 shows that the computation of $M_{\text{opt}}(\Pi, \mathcal{C})$ in (55) subsumes maximum-likelihood hard-decision decoding of Reed-Solomon codes. No polynomial-time algorithm for maximum-likelihood hard-decision decoding of Reed-Solomon codes is presently known [26], and the problem is generally considered to be hard.

Acknowledgement. We are grateful to David Forney, Marc Fossorier, Tom Høholdt, Elias Masry, Alon Orlitsky, and Madhu Sudan for stimulating discussions.

References

[1] D. Augot and L. Pecquet, “A Hensel lifting to replace factorization in list-decoding of algebraic-geometric and Reed-Solomon codes,” *IEEE Trans. Inform. Theory*, vol. 46, pp. 2605–2614, November 2000.

[2] E.R. Berlekamp, “Bounded distance+1 soft-decision Reed-Solomon decoding,” *IEEE Trans. Inform. Theory*, vol. 42, pp. 704–721, May 1996.

[3] E.R. Berlekamp, R.J. McEliece, and H.C.A. van Tilborg, “On the inherent intractability of certain coding problems,” *IEEE Trans. Inform. Theory*, vol. 24, pp. 384–386, May 1978.

[4] E.R. Berlekamp, R.E. Peile, and S.P. Pope, “The application of error control to communications,” *IEEE Commun. Mag.*, vol. 25, pp. 44–57, January 1987.

[5] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, New York: Wiley Interscience, 1991.

[6] A.B. Cooper, 3-rd, “Soft-decision decoding of Reed-Solomon codes,” pp. 108–124, in S.B. Wicker and V.K. Bhargava, Editors, *Reed-Solomon Codes and their Applications*, New York: IEEE Press, 1994.

[7] G.-L. Feng, “A fast special factorization algorithm in the Sudan decoding procedure,” in *Proc. 31-th Allerton Conf. Comm., Contr., and Computing*, September 2000.

[8] G.-L. Feng, “Very fast algorithms in Sudan decoding procedure for Reed-Solomon codes,” *IEEE Trans. Inform. Theory*, submitted for publication, December 2000.

[9] G.D. Forney, Jr., “Generalized minimum distance decoding,” *IEEE Trans. Inform. Theory*, vol. 12, pp. 125–131, April 1966.

[10] G.D. Forney, Jr., *Concatenated Codes*, Cambridge, MA: MIT Press, 1966.

[11] V. Guruswami and M. Sudan, “Improved decoding of Reed-Solomon and algebraic-geometric codes,” *IEEE Trans. Inform. Theory*, vol. 45, pp. 1755–1764, 1999.

[12] V. Guruswami and M. Sudan, “List decoding algorithms for certain concatenated codes,” *IEEE Trans. Inform. Theory*, submitted for publication, December 2000.

[13] K.A.S. Immink, *Coding Techniques for Digital Recorders*, Englewood Cliffs, NJ: Prentice-Hall, 1991.

[14] R. Kötter, *On Algebraic Decoding of Algebraic-Geometric and Cyclic codes*, Ph.D. Thesis, University of Linköping, Sweden, 1996.

[15] R. Kötter, “Fast generalized minimum distance decoding of algebraic geometric and Reed-Solomon codes,” *IEEE Trans. Inform. Theory*, vol. 42, pp. 721–738, May 1996.

[16] R.J. McEliece and L. Swanson, "Reed-Solomon codes and the exploration of the solar system," pp. 25–40, in S.B. Wicker and V.K. Bhargava, Editors, *Reed-Solomon Codes and their Applications*, New York: IEEE Press, 1994.

[17] R.R. Nielsen, "Decoding concatenated codes with Sudan's algorithm," preprint, July 2001.

[18] R.R. Nielsen and T. Høholdt, "Decoding Reed-Solomon codes beyond half the minimum distance," preprint, 1998; see also `http://www.student.dtu.dk/ p938546`.

[19] V. Olshevsky and A. Shokrollahi, "A displacement structure approach to efficient decoding of Reed-Solomon and algebraic-geometric codes," in *Proceedings.* 31-*th ACM Symp. Theory of Computing* (STOC), pp. 235–244, Atlanta, GA., May 1999.

[20] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, New York: McGraw-Hill, 1984.

[21] V. Ponnampalam and B.S. Vucetic, "Soft decision decoding of Reed-Solomon codes," in *Proc.* 13-*th Symp. Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes*, Honolulu, HI., USA, November 1999.

[22] S. Ray-Chaudhuri and A.H. Chan, "Bit-level parallel decoding of Reed-Solomon codes," in *Proc.* 31-*th Allerton Conf. Comm., Contr., and Computing*, September 1993.

[23] R.M. Roth and G. Ruckenstein, "Efficient decoding of Reed-Solomon codes beyond half the minimum distance", *IEEE Trans. Inform. Theory*, vol. 46, pp. 246–258, January 2000.

[24] U. Sorger, "A new Reed-Solomon decoding algorithm based on Newton's interpolation," *IEEE Trans. Inform. Theory*, vol. 39, pp. 358–365, March 1993.

[25] M. Sudan, "Decoding of Reed-Solomon codes beyond the error correction bound," *J. Complexity*, vol. 12, pp. 180–193, 1997.

[26] A. Vardy, "Algorithmic complexity in coding theory and the minimum distance problem," in *Proc.* 29-*th Symp. Theory of Computing*, pp. 92–109, El Paso, TX., 1997.

[27] A. Vardy and Y. Be'ery, "Bit-level soft-decision decoding of Reed-Solomon codes," *IEEE Trans. Commun.*, vol. 39, pp. 440–445, March 1991.

[28] L.R. Welch and E.R. Berlekamp, *Error correction for algebraic block codes*, U.S. Patent No. 4,633,470, issued December 30, 1986.

[29] W.W. Wu, D. Haccoun, R.E. Peile, and Y. Hirata, "Coding for satellite communication," *IEEE J. Select. Areas Commun.*, vol. 5, pp. 724-785, 1987.

[30] X-W. Wu and P.H. Siegel, "Efficient root-finding algorithm with application to list decoding of algebraic-geometric codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 2579–2587, September 2001.