Simple MAP Decoding of First-Order Reed–Muller and Hamming Codes

Alexei Ashikhmin, *Member, IEEE,* and
Simon Litsyn, *Senior Member, IEEE*

***Abstract*—A maximum *a posteriori* (MAP) probability decoder of a block code minimizes the probability of error for each transmitted symbol separately. The standard way of implementing MAP decoding of a linear code is the Bahl–Cocke–Jelinek–Raviv (BCJR) algorithm, which is based on a trellis representation of the code. The complexity of the BCJR algorithm for the first-order Reed–Muller (RM-1) codes and Hamming codes is proportional to n^2, where n is the code's length. In this correspondence, we present new MAP decoding algorithms for binary and nonbinary RM-1 and Hamming codes. The proposed algorithms have complexities proportional to $q^2 n \log_q n$, where q is the alphabet size. In particular, for the binary codes this yields complexity of order $n \log n$.**

***Index Terms*—Hamming codes, maximum *a posteriori* (MAP) decoding, Reed–Muller codes.**

I. Introduction

The first-order Reed–Muller (RM-1) codes is a family of low-rate linear codes with the maximal possible minimum distance for this rate and length. They find numerous applications because of their optimal parameters and simple decoding algorithms. Many other codes of higher rates can be seen as a union of cosets of the RM-1 codes. Therefore, a decoding of the RM-1 codes can be employed in their decoding as well. (Extended) Hamming codes are dual to RM-1 codes. They also possess optimal parameters, and have even more numerous applications, e.g., in product code schemes.

Decoding algorithms for RM-1 and Hamming codes were considered in many papers and monographs. For an introduction to maximum-likelihood (ML) decoding of binary RM-1 codes see, e.g., [11], [16], [17]. Decoding algorithms for binary RM-1 codes guaranteeing error correction of up to half of the minimum (Euclidean or Hamming) distance can be found in [1]–[3], [18]. Suboptimal decoding of nonbinary RM-1 codes was considered in [4], [13]. Hard-decision ML decoding of the Hamming codes is the trivial syndrome decoding. For soft-decision ML decoding of the Hamming codes see, e.g., [10], [16], [17], [20], [25], [27], and for suboptimal decoding see, e.g., [21], [26].

In this correspondence, we address the complexity of maximum *a posteriori* (MAP) decoding of RM-1 and Hamming codes. A MAP decoder of a linear code computes *a posteriori* probabilities for all possible values of each transmitted symbol and chooses the maximum one. A standard way of implementing this is the Bahl–Cocke–Jelinek–Raviv (BCJR) algorithm [5]. This algorithm is based on a trellis representation of the code. Such a representation allows one to reduce complexity of MAP decoding in comparison with the straightforward computation, though it still remains prohibitively complicated for most of practically interesting codes.

Manuscript received May 29, 2002; revised March 19, 2004. This work was supported in part by a grant from ISF. The material in this correspondence was presented in part at the Conference on Applied Mathematics, University of Central Oklahoma, Oklahoma, OK, October 2002 and at the IEEE Information Theory Workshop, Paris, France, March/April 2003.

A. Ashikhmin is with Bell Laboratories, Lucent Technologies, Murray Hill, NJ 07974 USA (e-mail: aea@research.bell-labs.com).

S. Litsyn is with the Department of Electrical Engineering–Systems, Tel-Aviv University, Ramat-Aviv, 69978 Israel (e-mail: litsyn@eng.tau.ac.il).

Communicated by J. Justesen, Associate Editor for Coding Theory.

Digital Object Identifier 10.1109/TIT.2004.831835

The BCJR algorithm for binary RM-1 and Hamming codes has complexity proportional to n^2, where n is the code length, see, e.g., [16], [17]. To the extent of our knowledge, there are no results on the complexity of BCJR decoding for nonbinary RM-1 and Hamming codes, though we believe that it is proportional to $q^2 n^2$. Some approaches to suboptimal decoding of Hamming codes approximating MAP were considered, e.g., in [8], [14], [23]. An approximated MAP decoding for RM-1 was considered in [19].

In this correspondence, we propose a MAP decoding algorithm for both RM-1 and Hamming codes, with smaller complexity than the one of BCJR. The algorithm is based on fast Hadamard transforms and requires the number of operations proportional to $q^2 n \log_q n$. The correspondence is organized as follows. In Section II, we give a formal definition of the MAP decoding of a linear code. In Sections III and IV, we present the new soft-input soft-output (SISO) MAP decoding algorithms for RM-1 and Hamming codes, respectively. In Section V, we proceed with a description of MAP decoding of a nonbinary linear code in the language of a group algebra. In Sections VI and VII, we present SISO MAP decoding algorithms for nonbinary RM-1 and Hamming codes.

II. MAP Decoding of a Linear Code

For the sake of completeness, we give a description of MAP decoding of an arbitrary linear code. Let $\mathbb{F}_q$ be the Galois field of size q, and let C be an $[n, k]$ linear code over $\mathbb{F}_q$. We assume that we transmit codewords of C through a memoryless channel. Let Q be an output sampling space, either finite or infinite, of the channel. In the case of finite Q, we assume that the conditional probabilities $\Pr(y|\alpha)$ of receiving $y \in Q$ under the condition that $\alpha \in \mathbb{F}_q$ has been transmitted are known. If Q is infinite, then y is a continuous random variable with probability distribution function (p.d.f.), say $f_\alpha(y)$, defined by the transmitted symbol $\alpha \in \mathbb{F}_q$. For instance, if we use a binary phase-shift keying (BPSK) modulated binary code in a channel with additive white Gaussian noise, then we have two p.d.f's

$$f_0(y) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{(x-1)^2}{2\sigma^2}} \quad \text{and} \quad f_1(y) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{(x+1)^2}{2\sigma^2}}$$

where $\sigma = \frac{n}{2k\vartheta}$ depends on the channel signal-to-noise ratio ϑ. We will assume that the functions $f_\alpha(y)$ are known for all $\alpha \in \mathbb{F}_q$.

For the rest of the correspondence, we will assume that Q is finite and we will work with the conditional probabilities $\Pr(y_i|\alpha)$. In the case of infinite Q, one has to replace the conditional probabilities with the density functions $f_\alpha(y_j)$.

Since we assume a memoryless channel, the conditional probability of receiving a vector $\underline{y} \in Q^n$ under the condition that a vector $\underline{c} \in \mathbb{F}_q^n$ has been transmitted can be written as the product

$$\Pr(\underline{y}|\underline{c}) = \prod_{j=0}^{n-1} \Pr(y_j|c_j). \tag{1}$$

We also assume that all codewords have the same probability $\frac{1}{q^k}$ to be transmitted.

Let $\underline{c} = (c_0, c_1, \ldots, c_{n-1}) \in C$ be transmitted, and the received vector be $\underline{y} \in Q^n$. Our goal is to determine what was the jth transmitted symbol. To do this, we estimate the conditional probabilities $\Pr(c_j = \alpha|\underline{y})$ for all $\alpha \in \mathbb{F}_q$ and choose the maximum one.

Let

$$C_j^{(\alpha)} = \{\underline{c} = (c_0, c_1, \ldots, c_{n-1}) \in C : c_j = \alpha\}$$

0018-9448/04$20.00 © 2004 IEEE

i.e., $C_j^{(\alpha)}$ is the set of all codewords whose jth symbol is α. Then in the case of finite Q, using the Bayes rule and (1), we can write

$$\begin{aligned}\Pr(c_j=\alpha|\underline{y}) &= \sum_{\underline{b}\in C_j^{(\alpha)}} \Pr(\underline{c}=\underline{b}|\underline{y}) \\ &= \sum_{\underline{b}\in C_j^{(\alpha)}} \frac{\Pr(\underline{y}|\underline{b})}{\Pr(\underline{y})}\Pr(\underline{b}) \\ &= \frac{1}{q^k\Pr(\underline{y})}\sum_{\underline{b}\in C_j^{(\alpha)}}\prod_{l=0}^{n-1}\Pr(y_l|b_l). \qquad (2)\end{aligned}$$

The denominator of (2) is the same for any $\alpha\in\mathbb{F}_q$. Therefore , for MAP decoding it is enough to estimate only the sum of products of $\Pr(y_i|b_j)$ for different $\alpha\in\mathbb{F}_q$ and to choose the maximum one.

In what follows, we will denote by the symbols $\bullet$, $\oplus$, and $\ominus$, the multiplication, addition, and subtraction in $\mathbb{F}_q$, respectively. To simplify notation, we will use integer numbers and indices, such as i, j, and so on, either as natural numbers or as elements of $\mathbb{F}_q$. Their meaning will be clear from the context. Through the correspondence, we will assume that with an $[n, k+1]$ code, say C, containing the all-one codeword $\mathbf{1}$, a $2^k\times n$ matrix D is associated. It is constructed as follows. Let $\widehat{C}$ be an $[n,k]$ subcode of C not containing $\mathbf{1}$. We will enumerate the codewords of C and say that $\underline{c}_i$ is the ith codeword of C, and c_{ij} is the jth symbol of the ith codeword. We also assume that the codewords with indices from 0 to 2^k-1 form the subcode $\widehat{C}$. In the case of binary codes, we will define D as the matrix whose ith row is the ith codeword of $\widehat{C}$ with zeros replaced by ones and ones replaced by minus ones. The matrix D for nonbinary codes will be defined in Section V.

III. Binary Reed–Muller Codes

We start with basic definitions. The binary first-order Reed-Muller (RM-1) codes $\mathcal{RM}(1,m)$ are $[2^m, m+1, 2^{m-1}]$ linear codes. Let $H_{2^m}=[h_{i,j}]$ be the $2^m\times 2^m$ Hadamard matrix, where $h_{i,j}=(-1)^{i_0j_0\oplus\cdots\oplus i_{m-1}j_{m-1}}$, and $i_0,\ldots,i_{m-1}$ as well as $j_0,\ldots,j_{m-1}$ are binary representations of the indices i and j, respectively. Then all codewords of $\mathcal{RM}(1,m)$, with zeros and ones replaced by ones and minus ones, respectively, are exactly the rows of H_{2^m} and their negations. In other words, H_{2^m} can be chosen as the matrix D of $\mathcal{RM}(1,m)$.

Let $\underline{v}=(v_0,\ldots,v_{n-1})$ be a real vector. The fast Hadamard transform, which we denote by $\mathbf{FHT}_{\mathbb{R}}$, requires only $n\log_2 n$ operations for computing the product $\underline{v}H_n$. In what follows, for the sake of completeness, we give a formal description of this transform. To compute $(s_0,\ldots,s_{n-1})=\mathbf{FHT}_{\mathbb{R}}(v_0,\ldots,v_{n-1})$ perform the following steps $\log_2 n$ times.

For j from 0 to $n/2-1$ compute:

$$s_j=v_{2j}+v_{2j+1},\quad s_{n/2+j}=v_{2j}-v_{2j+1}.$$

For j from 0 to $n-1$ assign: $v_j=s_j$.

The idea of MAP decoding of $\mathcal{RM}(1,m)$ is based on twofold use of $\mathbf{FHT}_{\mathbb{R}}$ transform. First, we use this transform for fast computation of probabilities $\Pr(\underline{y}|\underline{c}_i)$. Next, after some auxiliary computations, we use this transform for fast computation of soft decisions $\Pr(c_j=0|\underline{y})-\Pr(c_j=1|\underline{y})$ for all transmitted bits.

The algorithm starts with computation of the vector $\underline{b}=(b_0,\ldots,b_{n-1})$, where

$$b_j=\ln\Pr(y_j|0)-\ln\Pr(y_j|1),\qquad j=0,\ldots,n-1.$$

Next, we notice that if C is an $[n,k+1]$ code and C contains the all-one codeword then entries of the vector $\underline{t}=(t_0,\ldots,t_{2^k-1})=\underline{b}D^{\perp}$ are equal to the following expression:

$$t_i=\sum_{j=0}^{n-1}\ln\Pr(y_j|c_{ij})-\sum_{j=0}^{n-1}\ln\Pr(y_j|c_{ij}\oplus 1),\quad i=0,\ldots,2^k-1.$$

As we have mentioned before, in the case of $\mathcal{RM}(1,m)$ code $D=H_{2^m}$. Hence, we can compute the vector $\underline{t}$ with the help of $\mathbf{FHT}_{\mathbb{R}}$ transform as follows: $\underline{t}=(t_0,\ldots,t_{2^m-1})=\mathbf{FHT}_{\mathbb{R}}(b_0,\ldots,b_{n-1})$. We use the following simple trick to compute the sums of logarithms of probabilities from the differences of the sums. Let

$$s=\sum_{j=0}^{n-1}\ln\Pr(y_j|0)+\ln\Pr(y_j|1).$$

Then

$$r_i^{(0)}=\frac{s+t_i}{2}=\sum_{j=0}^{n-1}\ln\Pr(y_j|c_{ij})$$

and

$$r_i^{(1)}=\frac{s-t_i}{2}=\sum_{j=0}^{n-1}\ln\Pr(y_j|c_{ij}\oplus\mathbf{1}).$$

With the help of exponentiation, we recompute the sums of logarithms of probabilities into products of the probabilities as follows:

$$v_i^{(0)}=e^{r_i^{(0)}}=\prod_{j=0}^{n-1}\Pr(y_j|c_{ij})=\Pr(\underline{y}|\underline{c}_i)$$

and

$$v_i^{(1)}=e^{r_i^{(1)}}=\prod_{j=0}^{n-1}\Pr(y_j|c_{ij}\oplus\mathbf{1})=\Pr(\underline{y}|\underline{c}_i\oplus\mathbf{1}).$$

This finishes the first part of the algorithm. The second part of the algorithm is concerned with computing soft decisions $\Pr(c_j=0|\underline{y})-\Pr(c_j=1|\underline{y})$. Let us notice that if C is an $[n,k+1]$ code that contains the all-one codeword and

$$x_i=\Pr(\underline{y}|\underline{c}_i)-\Pr(\underline{y}|\underline{c}_i\oplus\mathbf{1}),\qquad i=0,\ldots,2^k-1$$

then the entries of the vector $(w_0,\ldots,w_{n-1})=(x_0,\ldots,x_{2^k-1})D$ are equal to

$$\begin{aligned}w_j&=\sum_{i=0}^{2^k-1}d_{ij}\Big(\Pr(\underline{y}|\underline{c}_i)-\Pr(\underline{y}|\underline{c}_i\oplus\mathbf{1})\Big)\\ &=\sum_{\underline{c}\in C_j^{(0)}}\Pr(\underline{y}|\underline{c})-\sum_{\underline{c}\in C_j^{(1)}}\Pr(\underline{y}|\underline{c}),\qquad j=0,\ldots,n-1. \qquad (3)\end{aligned}$$

Since in our case $D=H_{2^m}$ we can use the $\mathbf{FHT}_{\mathbb{R}}$ transform for these computations. First, we form a vector $\underline{x}$ with the entries $x_i=v_i^{(0)}-v_i^{(1)}$, and next compute $(w_0,w_1,\ldots,w_{n-1})=\mathbf{FHT}_{\mathbb{R}}(x_0,x_1,\ldots,x_{n-1})$. From (3) it follows that

$$\begin{aligned}w_j&=\sum_{\underline{c}\in\mathcal{RM}(1,m)_j^{(0)}}\Pr(\underline{y}|\underline{c})-\sum_{\underline{c}\in\mathcal{RM}(1,m)_j^{(1)}}\Pr(\underline{y}|\underline{c})\\ &=2^{m+1}\Pr(\underline{y})\left(\Pr(c_j=0|\underline{y})-\Pr(c_j=1|\underline{y})\right).\end{aligned}$$

The constant factor $2^{m+1}\Pr(\underline{y})$ can be computed as follows:

$$\begin{aligned}q&=\sum_{i=0}^{2^m-1}\left(v_i^{(0)}+v_i^{(1)}\right)\\ &=\sum_{\underline{c}\in\mathcal{RM}(1,m)_j^{(0)}}\Pr(\underline{y}|\underline{c})+\sum_{\underline{c}\in\mathcal{RM}(1,m)_j^{(1)}}\Pr(\underline{y}|\underline{c})\\ &=2^{m+1}\Pr(\underline{y})\left[\Pr(c_j=0|\underline{y})+\Pr(c_j=1|\underline{y})\right]\\ &=2^{m+1}\Pr(\underline{y}).\end{aligned}$$

Finally, we compute the soft decisions for all code bits as follows:

$$\frac{w_j}{q} = \Pr(c_j = 0|\underline{y}) - \Pr(c_j = 1|\underline{y}), \qquad j = 0, \ldots, n-1.$$

Since the complexity of the $\mathbf{FHT}_{\mathbb{R}}$ transform is $n \log_2 n$ and all other steps of the algorithm have linear complexity, the total complexity of the algorithm is $O(n \log_2 n)$.

IV. Binary Hamming Codes

Let C be an $[n, k]$ linear code over $\mathbb{F}_p$, p is a prime, and let $C^{\perp}$ be its dual. We assume that the codewords of $C^{\perp}$ are enumerated, and we denote them by $\underline{c}'$, i.e., c'_{ij} is the jth element of the ith codeword of $C^{\perp}$.

In [9], Hartmann and Rudolph developed an algorithm for the MAP decoding of a linear code using its dual code. In particular, they proved the following theorem.

Theorem 1: [9]

$$\Pr(c_j = s|\underline{y}) = \frac{1}{p^{n+1} \Pr(\underline{y})} A_j(s), \qquad s \in \mathbb{F}_p \tag{4}$$

where

$$A_j(s) = \sum_{t=0}^{p-1} \omega^{-st} \sum_{i=0}^{p^{n-k}-1} \left[\prod_{l=0}^{n-1} \sum_{r=0}^{p-1} \omega^{-r(c'_{il} \ominus t\delta_{jl})} \Pr(y_l|r) \right]$$

and

$$\omega = e^{2\pi\iota/p}, \quad \iota = \sqrt{-1}. \tag{5}$$

□

In the binary case, this equation becomes

$$A_j(s) = \sum_{t=0}^{1} (-1)^{st} \sum_{i=0}^{2^{n-k}-1} \prod_{l=0}^{n-1} \Big[\Pr(y_l|0) + (-1)^{(c'_{il} \oplus t\delta_{jl})} \Pr(y_l|1) \Big]. \tag{6}$$

Let

$$\psi_j = \Pr(y_j|1)/\Pr(y_j|0)$$

and

$$\begin{aligned} \rho_j &= \frac{1 - \Pr(y_j|1)/\Pr(y_j|0)}{1 + \Pr(y_j|1)/\Pr(y_j|0)} \\ &= \Pr(c_j = 0|y_j) - \Pr(c_j = 1|y_j). \end{aligned}$$

It was noted in [9] that dividing (6) by $\prod_{j=0}^{n-1} \Pr(y_j|0)$ and further by $\prod_{j=0}^{n-1}(1 + \psi_j)$, and using identity

$$\frac{1 + \psi_j(-1)^{c'_{ij} \oplus \delta_{tj}}}{1 + \psi_j} = \left(\frac{1 - \psi_j}{1 + \psi_j} \right)^{c'_{ij} \oplus \delta_{tj}}$$

one obtains

$$A_j(0) - A_j(1) = \lambda \sum_{i=0}^{2^{n-k}-1} \prod_{l=0}^{n-1} \rho_l^{c'_{il} \oplus \delta_{jl}}. \tag{7}$$

In a similar way, one can obtain

$$A_j(0) + A_j(1) = \lambda \sum_{i=0}^{2^{n-k}-1} \prod_{l=0}^{n-1} \rho_l^{c'_{il}}. \tag{8}$$

From (7) and (8) it follows that

$$\begin{aligned} &\Pr(c_j = 0|\underline{y}) - \Pr(c_j = 1|\underline{y}) \\ &\quad = \frac{\Pr(c_j = 0|\underline{y}) - \Pr(c_j = 1|\underline{y})}{\Pr(c_j = 0|\underline{y}) + \Pr(c_j = 1|\underline{y})} \\ &\quad = \frac{\sum_{i=0}^{2^{n-k}-1} \prod_{l=0}^{n-1} \rho_l^{c'_{il} \oplus \delta_{jl}}}{\sum_{i=0}^{2^{n-k}-1} \prod_{l=0}^{n-1} \rho_l^{c'_{il}}}. \end{aligned} \tag{9}$$

We will present an algorithm for fast computing the above expression in the case of the extended Hamming code $\mathcal{H}_m$ Hamming code. It is an $[2^m, 2^m - m - 1, 4]$ code, and can be seen as the dual code of $\mathcal{RM}(1, m)$.

Let

$$\left(\delta_0^{(0)}, \delta_1^{(0)}, \ldots, \delta_{n-1}^{(0)}; \delta_0^{(1)}, \delta_1^{(1)}, \ldots, \delta_{n-1}^{(1)} \right)$$

and

$$\left(\beta_0^{(0)}, \beta_1^{(0)}, \ldots, \beta_{n-1}^{(0)}; \beta_0^{(1)}, \beta_1^{(1)}, \ldots, \beta_{n-1}^{(1)} \right)$$

be binary vectors. In what follows, we will need the following Walsh–Hadamard-type transform, which we denote by $\mathbf{FWH}_{\mathcal{F}_2[\mathbb{Z}_2]}$. The reason for the notation will be clarified in Section VII. To compute

$$\left(\delta_0^{(0)}, \ldots, \delta_{n-1}^{(0)}, \delta_0^{(1)}, \ldots, \delta_{n-1}^{(1)} \right) = \mathbf{FWH}_{\mathcal{F}_2[\mathbb{Z}_2]} \left(\beta_0^{(0)}, \ldots, \beta_{n-1}^{(0)}, \beta_0^{(1)}, \ldots, \beta_{n-1}^{(1)} \right)$$

perform the following steps $\log_2 n$ times:

For j from 0 to $n/2 - 1$ compute

$$\delta_j^{(0)} = \beta_{2j}^{(0)} \oplus \beta_{2j+1}^{(0)}, \quad \delta_j^{(1)} = \beta_{2j}^{(1)} \oplus \beta_{2j+1}^{(1)}$$
$$\delta_{n/2+j}^{(0)} = \beta_{2j}^{(0)} \oplus \beta_{2j+1}^{(1)}, \quad \delta_{n/2+j}^{(0)} = \beta_{2j}^{(1)} \oplus \beta_{2j+1}^{(0)}.$$

For j from 0 to $n - 1$ assign $\beta_j^{(0)} = \delta_j^{(0)}$ and $\beta_j^{(1)} = \delta_j^{(1)}$.

Let $\mathcal{I}_i^{(0)}$ and $\mathcal{I}_i^{(1)}$ be the sets of positions where the ith codeword of $\mathcal{RM}(1, m)$ has 0's and 1's, respectively. The idea of MAP decoding of the Hamming codes is first to use $\mathbf{FWH}_{\mathbb{R}}$ and $\mathbf{FWH}_{\mathcal{F}_2[\mathbb{Z}_2]}$ transforms for fast computing $\prod_{j \in \mathcal{I}_i^{(0)}} \rho_j$ and $\prod_{j \in \mathcal{I}_i^{(1)}} \rho_j$, and, second, to use these expressions and $\mathbf{FWH}_{\mathbb{R}}$ for fast computing soft decisions $\Pr(c_j = 0|\underline{y}) - \Pr(c_j = 1|\underline{y})$ for all code bits according to (9).

The algorithm starts with computing

$$\rho_j = \frac{1 - \frac{\Pr(y_j|1)}{\Pr(y_j|0)}}{1 + \frac{\Pr(y_j|1)}{\Pr(y_j|0)}} = \Pr(c_j = 0|y_j) - \Pr(c_j = 1|y_j)$$

and

$$\tau_j = \ln|\rho_j|, \qquad j = 0, \ldots, n-1.$$

Next we encode the signs of ρ_j into pairs of bits as follows:

$$\beta_j^{(0)} = \begin{cases} 0, & \text{if } \rho_j \geq 0 \\ 1, & \text{if } \rho_j < 0 \end{cases}$$

and

$$\beta_j^{(1)} = 0, \qquad j = 0, \ldots, n-1.$$

Further, we compute the vector

$$\underline{s} = (s_0, s_1, \ldots, s_{2^m-1}) = \mathbf{FWH}_{\mathbb{R}}(\tau_0, \tau_1, \ldots, \tau_{n-1}).$$

From the definition of $\mathbf{FWH}_{\mathbb{R}}$ one can easily see that the entries of $\underline{s}$ are equal to

$$s_i = \sum_{j \in \mathcal{I}_i^{(0)}} \ln|\rho_j| - \sum_{j \in \mathcal{I}_i^{(1)}} \ln|\rho_j|, \qquad i = 0, \ldots, 2^m - 1.$$

Noting that $s_0 = \sum_{j=0}^{n-1} \ln|\rho_j|$, we obtain

$$t_i^{(0)} = \frac{s_0 + s_i}{2} = \sum_{j \in \mathcal{I}_i^{(0)}} \ln|\rho_j|$$

and

$$t_i^{(1)} = \frac{s_0 - s_i}{2} = \sum_{j \in \mathcal{I}_i^{(1)}} \ln|\rho_j|, \qquad i = 0, \ldots, 2^m - 1.$$

We cannot directly recompute the values of t_i into the products of ρ_j, since t_i do not contain any information about the signs. To reconstruct the signs we use the $\mathbf{FWH}_{\mathcal{F}_2[\mathbb{Z}_2]}$ transform and compute

$$\left(\delta_0^{(0)}, \ldots, \delta_{n-1}^{(0)}, \delta_0^{(1)}, \ldots, \delta_{n-1}^{(1)} \right) = \mathbf{FWH}_{\mathcal{F}_2[\mathbb{Z}_2]} \left(\beta_0^{(0)}, \ldots, \beta_{n-1}^{(0)}, \beta_0^{(1)}, \ldots, \beta_{n-1}^{(1)} \right).$$

From the definition of $\mathbf{FWH}_{\mathcal{F}_2[\mathbb{Z}_2]}$ one can see that

$$\delta_i^{(0)} = \sum_{j \in \mathcal{I}_i^{(0)}} \oplus \beta_j^{(0)} \text{ and } \delta_i^{(1)} = \sum_{j \in \mathcal{I}_i^{(1)}} \oplus \beta_j^{(0)}, \quad i = 0, \ldots, 2^m - 1.$$

Thus, we can compute products

$$h_i^{(0)} = \prod_{j \in \mathcal{I}_i^{(0)}} \rho_j \text{ and } h_i^{(1)} = \prod_{j \in \mathcal{I}_i^{(1)}} \rho_j$$

as follows:

$$h_i^{(0)} = \begin{cases} e^{t_i^{(0)}}, & \text{if } \delta_i^{(0)} = 0 \\ -e^{t_i^{(0)}}, & \text{if } \delta_i^{(0)} = 1 \end{cases}$$

and

$$h_i^{(1)} = \begin{cases} e^{t_i^{(1)}}, & \text{if } \delta_i^{(1)} = 0 \\ -e^{t_i^{(1)}}, & \text{if } \delta_i^{(1)} = 1 \end{cases} \qquad i = 0, \dots, 2^m - 1.$$

This finishes the first part of the algorithm.

The second part is concerned with computing soft decisions of the code bits. Let $x_i = h_i^{(0)} - h_i^{(1)}$, $i = 0, \dots, 2^m - 1$. Using arguments similar to ones used in Section II, one can see that the entries of $(r_0, r_1, \dots, r_{n-1}) = \mathbf{FWH}_{\mathbb{R}}(x_0, x_1, \dots, x_{2^m-1})$ are equal

$$r_j = \sum_{\underline{c}_t \in \mathcal{RM}(1,m)_j^{(0)}} \left[\prod_{l \in \mathcal{I}_t^{(0)}} \rho_l - \prod_{l \in \mathcal{I}_t^{(1)}} \rho_l \right] + \sum_{\underline{c}_t \in \mathcal{RM}(1,m)_j^{(1)}} \left[\prod_{l \in \mathcal{I}_t^{(1)}} \rho_l - \prod_{l \in \mathcal{I}_t^{(0)}} \rho_l \right].$$

Taking into account the obvious equality

$$q = \sum_{i=0}^{n-1} \left(h_i^{(0)} + h_i^{(1)} \right) = \sum_{\underline{c}_i \in \mathcal{RM}(1,m)} \left[\prod_{j \in \mathcal{I}_i^{(0)}} \rho_j + \prod_{j \in \mathcal{I}_i^{(1)}} \rho_j \right]$$

we obtain that

$$w_j^{(0)} = \frac{q - r_j}{2} = \sum_{\underline{c}_t \in \mathcal{RM}(1,m)_j^{(0)}} \prod_{l \in \mathcal{I}_t^{(0)}} \rho_l, \text{ and}$$

$$w_j^{(1)} = \frac{q + r_j}{2} = \sum_{\underline{c}_t \in \mathcal{RM}(1,m)_j^{(1)}} \prod_{l \in \mathcal{I}_t^{(1)}} \rho_l.$$

Finally, we compute soft decisions for all code bits as follows:

$$z_j = \frac{w_j^{(0)} \rho_j + \frac{w_j^{(1)}}{\rho_j}}{w_j^{(0)} + w_j^{(1)}} = \frac{A_j(0) - A_j(1)}{A_j(0) + A_j(1)} = \Pr(c_j = 0 | \underline{y}) - \Pr(c_j = 1 | \underline{y}), \qquad j = 0, \dots, n-1.$$

To examine the complexity of the algorithm, we note that all the steps except computations with the help of transforms $\mathbf{FWH}_{\mathbb{R}}$ and $\mathbf{FWH}_{\mathcal{F}_2[\mathbb{Z}_2]}$ have linear complexity. Therefore, the overall complexity of the algorithm is $O(n \log_2 n)$.

Remark: Note that one can use this algorithm for decoding of an arbitrary high-rate linear code. Indeed, since the parity-check matrix of the (nonextended) Hamming code consists of all nonzero binary m-tuples, we can decode an $[n, n-t]$ code, C, as the $\mathcal{H}_t$ code substituting $\rho_j = 0$ on the positions j that do not appear as columns of the parity-check matrix of $\mathcal{H}_m$. Let, for example, C be the $[2^m, 2^m - tm - 1]$ extended Bose–Chaudhuri–Hocquenghem (BCH) code. Then we can decode it using the algorithm for the code $\mathcal{H}_{tm}$. The complexity of such a decoding is proportional to $n^t t \log_2 n$. Usually, if C has a large minimum distance its trellis has $2^{tm} = n^t$ states. Hence, the BCJR algorithm requires about n^{t+1} operations. Thus, the suggested algorithm allows one to reduce the complexity of the BCJR algorithm by a factor of about $\frac{n}{t \log_2 n}$.

V. Group Algebra Description of MAP Decoding

We will formulate SISO MAP decoding of nonbinary RM-1 and Hamming codes in the language of a group algebra.

Let x be a formal variable. The set of monomials $x^\alpha, \alpha \in \mathbb{F}_q$ form a multiplicative group $\mathcal{F}_q$. The group $\mathcal{F}_q$ and the additive group of $\mathbb{F}_q$ are isomorphic. Addition in $\mathbb{F}_q$ corresponds to multiplication in $\mathcal{F}_q$. All formal sums

$$\sum_{\alpha \in \mathbb{F}_q} a^{(\alpha)} x^\alpha, \qquad a^{(\alpha)} \in \mathbb{R}$$

form the group algebra $\mathcal{F}_q[\mathbb{R}]$. Additions and multiplications in $\mathcal{F}_q[\mathbb{R}]$ are defined as follows:

$$\sum_{\alpha \in \mathbb{F}_q} a^{(\alpha)} x^\alpha + \sum_{\alpha \in \mathbb{F}_q} b^{(\alpha)} x^\alpha = \sum_{\alpha \in \mathbb{F}_q} (a^{(\alpha)} + b^{(\alpha)}) x^\alpha,$$

$$\sum_{\alpha \in \mathbb{F}_q} a^{(\alpha)} x^\alpha \cdot \sum_{\beta \in \mathbb{F}_q} b^{(\beta)} x^\beta = \sum_{\alpha, \beta \in \mathbb{F}_q} a^{(\alpha)} b^{(\beta)} x^{\alpha \oplus \beta} = \sum_{\alpha \in \mathbb{F}_q} x^\alpha \sum_{\beta \in \mathbb{F}_q} a^{(\beta)} b^{(\alpha \ominus \beta)}.$$

We will use bold font to denote elements of $\mathcal{F}_q[\mathbb{R}]$. It is easy to see that

$$(\boldsymbol{v}\boldsymbol{w})\boldsymbol{z} = \boldsymbol{v}(\boldsymbol{w}\boldsymbol{z})$$

and

$$\boldsymbol{v}(\boldsymbol{w} + \boldsymbol{z}) = \boldsymbol{v}\boldsymbol{w} + \boldsymbol{v}\boldsymbol{z}, \qquad \boldsymbol{v}, \boldsymbol{w}, \boldsymbol{z} \in \mathcal{F}_q[\mathbb{R}]. \tag{10}$$

Let again C be an $[n, k]$ linear code over $\mathbb{F}_q$ code that contains the all-one codeword, and let $\widehat{C}$ be an $[n, k]$ subcode of C such that $\mathbf{1} \notin \widehat{C}$.

We define $q^k \times n$ matrices D and $\overline{D}$ associated with C as follows:

$$D = [d_{i,j}] = [x^{c_{i,j}}] \quad \text{and} \quad \overline{D} = [\overline{d}_{i,j}] = [x^{\ominus c_{i,j}}]. \tag{11}$$

For example, if $\widehat{C}$ is a code over $\mathbb{F}_3$ consisting of the following three vectors $(0,0,0), (1,1,2), (2,2,1)$, then the matrices D and $\overline{D}$ are

$$D = \begin{bmatrix} x^0 & x^0 & x^0 \\ x^1 & x^1 & x^2 \\ x^2 & x^2 & x^1 \end{bmatrix} \quad \text{and} \quad \overline{D} = \begin{bmatrix} x^0 & x^0 & x^0 \\ x^{\ominus 1} & x^{\ominus 1} & x^{\ominus 2} \\ x^{\ominus 2} & x^{\ominus 2} & x^{\ominus 1} \end{bmatrix}.$$

Note that the entries of D and $\overline{D}$ are from $\mathcal{F}_q[\mathbb{R}]$. We again assume that a codeword $\underline{c} \in C$ has been transmitted and a vector $\underline{y} \in Q^n$ has been received. Let us compute from the vector $\underline{y}$ the vector $\underline{\boldsymbol{w}} = (\boldsymbol{w}_0, \boldsymbol{w}_1, \dots, \boldsymbol{w}_{n-1}) \in \mathcal{F}_q[\mathbb{R}]^n$, where

$$\boldsymbol{w}_i = \sum_{\alpha \in \mathbb{F}_q} w_i^{(\alpha)} x^\alpha = \sum_{\alpha \in \mathbb{F}_q} \ln \Pr(y_i | \alpha) x^\alpha$$

i.e.,

$$w_i^{(\alpha)} = \ln \Pr(y_i | \alpha).$$

Let

$$\underline{\boldsymbol{t}}^\top = \begin{bmatrix} \boldsymbol{t}_0 \\ \boldsymbol{t}_1 \\ \vdots \\ \boldsymbol{t}_{q^{k-1}-1} \end{bmatrix} = \overline{D} \underline{\boldsymbol{w}}^\top = \overline{D} \begin{bmatrix} \boldsymbol{w}_0 \\ \boldsymbol{w}_1 \\ \vdots \\ \boldsymbol{w}_{n-1} \end{bmatrix}.$$

Note that all operations are in $\mathcal{F}_q[\mathbb{R}]$ and

$$\boldsymbol{t}_i = \sum_{\alpha \in \mathbb{F}_q} t_i^{(\alpha)} x^\alpha \in \mathcal{F}_q[\mathbb{R}].$$

Proposition 2: $t_i^{(\alpha)} = \ln \Pr(\underline{y} | \underline{c}_i \oplus (\alpha, \alpha, \dots, \alpha))$.

Proof :

$$t_i = \sum_{j=0}^{n-1} x^{\ominus c_{i,j}} \boldsymbol{w}_j = \sum_{j=0}^{n-1} x^{\ominus c_{i,j}} \sum_{\alpha \in \mathbb{F}_q} w_j^{(\alpha)} x^{\alpha}$$
$$= \sum_{\alpha \in \mathbb{F}_q} x^{\alpha} \sum_{j=0}^{n-1} w_j^{(\alpha \oplus c_{i,j})} = \sum_{\alpha \in \mathbb{F}_q} x^{\alpha} \sum_{j=0}^{n-1} \ln \Pr(y_j | \alpha \oplus c_{i,j})$$
$$= \sum_{\alpha \in \mathbb{F}_q} x^{\alpha} \ln \prod_{j=0}^{n-1} \Pr(y_j | \alpha \oplus c_{i,j})$$
$$= \sum_{\alpha \in \mathbb{F}_q} \ln \Pr(\underline{y} | \underline{c}_i \oplus (\alpha, \alpha, \ldots, \alpha)) x^{\alpha}. \qquad \square$$

Let us now form the vector

$$(\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{q^{k-1}-1}) \in \mathcal{F}_q[\mathbb{R}]^n$$

where

$$\boldsymbol{v}_i = \sum_{\alpha \in \mathbb{F}_q} v_i^{(\alpha)} x^{\alpha} = \sum_{\alpha \in \mathbb{F}_q} e^{t_i^{(\alpha)}} x^{\alpha}$$
$$= \sum_{\alpha \in \mathbb{F}_q} \Pr(\underline{y} | \underline{c}_i \oplus (\alpha, \alpha, \ldots, \alpha)) x^{\alpha}.$$

Let

$$\underline{s}^{\top} = \begin{bmatrix} \boldsymbol{s}_0 \\ \boldsymbol{s}_1 \\ \vdots \\ \boldsymbol{s}_{n-1} \end{bmatrix} = D^{\top} \underline{\boldsymbol{v}}^{\top} = D^{\top} \begin{bmatrix} \boldsymbol{v}_0 \\ \boldsymbol{v}_1 \\ \vdots \\ \boldsymbol{v}_{q^{k-1}-1} \end{bmatrix}.$$

Then we have the following proposition.

Proposition 3:

$$s_j^{(\alpha)} = \sum_{\underline{b} \in C_j^{(\alpha)}} \Pr(\underline{y} | \underline{b}), \qquad j = 0, \ldots, n-1.$$

Proof: The proof is similar to the proof of Proposition 2. $\square$

From the definitions (2) of MAP decoding and Propositions 2 and 3, it follows that the following algorithm is the MAP decoding of the code C.

(Step 1) Compute $w_j^{(\alpha)} = \ln \Pr(y_j | \alpha)$; $0 \le j \le n-1$; $\alpha \in \mathbb{F}_q$, and then (Step 2) compute $\underline{\boldsymbol{t}}^{\top} = \overline{D} \underline{\boldsymbol{w}}^{\top}$. (Step 3) Transform t_i's into exponential form $v_i^{(\alpha)} = e^{t_i^{(\alpha)}}$, $0 \le i \le q^{k-1}-1$; $\alpha \in \mathbb{F}_q$. (Step 4) Compute $\underline{\boldsymbol{s}}^{\top} = D \underline{\boldsymbol{v}}^{\top}$, and (Step 5) assign $c_j = \beta$, $0 \le j \le n-1$, where β is such that

$$s_j^{(\beta)} = \max_{\alpha \in \mathbb{F}_q} \left\{ s_j^{(\alpha)} \right\}.$$

Let us estimate the complexity of the algorithm. Let $\boldsymbol{w} = \sum_{\alpha \in \mathbb{F}_q} w^{(\alpha)} x^{\alpha}$ be an arbitrary element of $\mathcal{F}_q[\mathbb{R}]$. Let $\boldsymbol{y} = x^{\beta}$, $\beta \in \mathbb{F}_q$, that is, $\boldsymbol{y}$ is an element of $\mathcal{F}_q[\mathbb{R}]$ of the special form. Then the multiplication

$$\boldsymbol{y}\boldsymbol{w} = \sum_{\alpha \in \mathbb{F}_q} w^{(\alpha \ominus \beta)} x^{\alpha}$$

is just a permutation of terms in $\boldsymbol{w}$. For this reason, the complexity of such kind of multiplication is assumed to be nil. Let now $\boldsymbol{z} = x^{\gamma}$ and $(\boldsymbol{w}_0, \boldsymbol{w}_1) \in \mathcal{F}_q[\mathbb{R}]^2$. Then the multiplication of the vectors

$$(\boldsymbol{y}, \boldsymbol{z}) \begin{pmatrix} \boldsymbol{w}_0 \\ \boldsymbol{w}_1 \end{pmatrix} = \sum_{\alpha \in \mathbb{F}_q} w_0^{\alpha \ominus \beta} x^{\alpha} + \sum_{\alpha \in \mathbb{F}_q} w_1^{\alpha \ominus \gamma} x^{\alpha}$$
$$= \sum_{\alpha \in \mathbb{F}_q} (w_0^{\alpha \ominus \beta} + w_1^{\alpha \ominus \gamma}) x^{\alpha}$$

requires q summations of real numbers. In general, the multiplication of a vector $(\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{n-1}) \in \mathcal{F}_q[\mathbb{R}]^n$ whose entries have the form $\boldsymbol{y}_j = x^{\beta_j}$, $\beta_j \in \mathbb{F}_q$, by a vector $(\boldsymbol{w}_0, \ldots, \boldsymbol{w}_{n-1}) \in \mathcal{F}_q[\mathbb{R}]^n$ requires $(n-1)q$ real summations.

The matrices D and $\overline{D}$ are $q^{k-1} \times n$ matrices whose entries are of the form $x^{\beta}, \beta \in \mathbb{F}_q$. Hence, we have three steps of the algorithm which have linear in n complexity, and two steps of the algorithm which require $q^{k-1}q(n-1)$ summations of real numbers.

VI. MAP Decoding of Nonbinary RM-1 Codes

In this section, we will show that for the RM-1 codes the complexity of the algorithm from Section V can be significantly reduced.

The q-ary RM-1 codes $\mathcal{RM}_q(1, m)$ were defined in [22], [24]. $\mathcal{RM}_q(1, m)$ has the length $n = q^m$, the dimension $k = m+1$, and the minimum distance $d = q^m - q^{m-1}$.

The following formal definition of $\mathcal{RM}_q(1, m)$ is convenient for our purposes. Let us represent the indices $0 \le j \le q^m - 1$ and $0 \le i \le q^{m+1} - 1$ in the q-ary form

$$j = j_0 + j_1 q + \cdots + j_{m-1} q^{m-1}, \qquad 0 \le j_l \le q-1$$
$$i = i_0 + i_1 q + \cdots + i_{m-1} q^{m-1} + r q^m, \quad 0 \le i_l, \ r \le q-1.$$

Then the jth symbol of the ith codeword of $\mathcal{RM}_q(1, m)$ is defined as follows:

$$c_{i,j} = i_0 \bullet j_0 \oplus i_1 \bullet j_1 \oplus \cdots \oplus i_{m-1} \bullet j_{m-1} \oplus r.$$

Similar to Section V, we denote by $\widehat{\mathcal{RM}}_q(1, m)$ the $[q^m, m]$ linear subcode of $\mathcal{RM}_q(1, m)$ that does not contain the vector $\mathbf{1}$ and whose jth symbol of the ith codeword is defined as follows:

$$c_{i,j} = i_0 \bullet j_0 \oplus i_1 \bullet j_1 \oplus \cdots \oplus i_{m-1} \bullet j_{m-1}.$$

For example, if $q = 3$, $m = 2$, and $i = 7$ then

$$\underline{c}_7 = (0, 1, 2, 2, 0, 1, 1, 2, 0).$$

Let us denote by $\otimes$ the Kronecker product of matrices and let F be a $q \times q$ matrix with entries from $\mathcal{F}_q[\mathbb{R}]$. It is straightforward to see that the following theorem, which Good [12] proved for matrices over a field, is also valid for the matrices over $\mathcal{F}_q[\mathbb{R}]$.

Theorem 4: [12] $\underbrace{F \otimes F \otimes \cdots \otimes F}_{m \text{ times}} = A^m$, where the entries of A are

$$a_{i,j} = f_{i_0, j_{m-1}} \delta_{i_1, j_0} \delta_{i_2, j_1} \cdots \delta_{i_{m-1}, j_{m-2}}$$

and $i_0, \ldots, i_{m-1}$ as well as $j_0, \ldots, j_{m-1}$ are binary representations of i and j, respectively. $\square$

For example, if $q = 3$ and $m = 2$, then

$$A = \begin{bmatrix} f_{0,0} & f_{0,1} & f_{0,2} & & & & & & \\ & & & f_{0,0} & f_{0,1} & f_{0,2} & & & \\ & & & & & & f_{0,0} & f_{0,1} & f_{0,2} \\ f_{1,0} & f_{1,1} & f_{1,2} & & & & & & \\ & & & f_{1,0} & f_{1,1} & f_{1,2} & & & \\ & & & & & & f_{1,0} & f_{1,1} & f_{1,2} \\ f_{2,0} & f_{2,1} & f_{2,2} & & & & & & \\ & & & f_{2,0} & f_{2,1} & f_{2,2} & & & \\ & & & & & & f_{2,0} & f_{2,1} & f_{2,2} \end{bmatrix}.$$

Using Theorem 4, we define the following fast Hadamard transform $\mathbf{FHT}_{\mathcal{F}_q[\mathbb{R}]}$:

$$(\boldsymbol{s}_0, \ldots, \boldsymbol{s}_{n-1}) = \mathbf{FHT}_{\mathcal{F}_q[\mathbb{R}]}(\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{n-1}) :$$
$$\{\text{repeat } m \text{ times } \underline{\boldsymbol{s}} = A \underline{\boldsymbol{v}}, \underline{\boldsymbol{v}} = \underline{\boldsymbol{s}}.\}.$$

It is easy to see that this algorithm requires $qn \log_q n$ summations of real numbers.

Remark: The reason for the notation $\mathbf{FHT}_{\mathcal{F}_2[\mathbb{Z}_2]}$ in Section IV is that this transform can be naturally defined in terms of the group algebra $\mathcal{F}_2[\mathbb{Z}_2]$.

For an element $\boldsymbol{v} = \sum_{\alpha\in\mathbb{F}_q} a^{(\alpha)}x^{(\alpha)} \in \mathcal{F}_q[\mathbb{R}]$ we define $\overline{\boldsymbol{v}}$ as follows:

$$\overline{\boldsymbol{v}} = \sum_{\alpha\in\mathbb{F}_q} a^{(\alpha)}x^{(\ominus\alpha)} \in \mathcal{F}_q[\mathbb{R}].$$

Now we will define transform $\overline{\mathbf{FHT}}_{\mathcal{F}_q[\mathbb{R}]}(\cdot)$ as $\mathbf{FHT}_{\mathcal{F}_q[\mathbb{R}]}(\cdot)$ transform in which the matrix A is replaced by $\overline{A}$.

Let us define entries of F as follows: $f_{i,j} = x^{i\bullet j}, 0 \le i,j \le q-1$. Note that in this definition i and j are considered as elements of $\mathbb{F}_q$. If, for instance, $q = 3$

$$F = \begin{bmatrix} x^0 & x^0 & x^0 \\ x^0 & x^1 & x^2 \\ x^0 & x^2 & x^1 \end{bmatrix}.$$

Let $\overline{F} = [\overline{f}_{i,j}]$, where $\overline{f}_{i,j} = x^{\ominus(i\bullet j)}$. Let now D and $\overline{D}$ be the matrices defined by (11) and by the definition of $\widehat{\mathcal{RM}}_q(1,m)$.

Lemma 5:

$$D = \underbrace{F\otimes F\otimes\cdots\otimes F}_{m \text{ times}} \text{ and } \overline{D} = \underbrace{\overline{F}\otimes\overline{F}\otimes\cdots\otimes\overline{F}}_{m \text{ times}}.$$

Proof: Let

$$E = [e_{i,j}] = \underbrace{F\otimes F\otimes\cdots\otimes F}_{m \text{ times}}, \qquad 0 \le i,j \le q^m-1.$$

From the definition of the Kronecker product we have

$$e_{i,j} = \prod_{l=0}^{m-1} f_{i_l,j_l}$$

where

$$i = i_0q^{m-1} + i_1q^{m-2} + \cdots + i_{m-1}, \qquad 0 \le i_l \le q-1$$
$$j = j_0q^{m-1} + j_1q^{m-2} + \cdots + j_{m-1}, \qquad 0 \le j_l \le q-1.$$

Hence,

$$e_{i,j} = \prod_{l=0}^{m-1} f_{i_l,j_l} = \prod_{l=0}^{m-1} x^{i_l\bullet j_l}$$
$$= x^{i_0\bullet j_0\oplus i_1\bullet j_1\oplus\cdots\oplus i_{m-1}\bullet j_{m-1}} = x^{c_{i,j}} = d_{i,j}.$$

The proof of the second equation is similar. □

Now, taking into account (10), we can simplify the MAP algorithm described in Section V for the case of $\mathcal{RM}_q(1,m)$ codes. Namely, we can replace the second step with the computation

$$(\boldsymbol{z}_0,\ldots,\boldsymbol{z}_{n-1}) = \overline{\mathbf{FHT}}_{\mathcal{F}_q[\mathbb{R}]}(\boldsymbol{w}_0,\ldots,\boldsymbol{w}_{n-1})$$

and the fourth step with the computation

$$(\boldsymbol{v}_0,\ldots,\boldsymbol{v}_{n-1}) = \mathbf{FHT}_{\mathcal{F}_q[\mathbb{R}]}(\boldsymbol{z}_0,\ldots,\boldsymbol{z}_{n-1}).$$

It is easy to see that the complexity of this algorithm is proportional to $qn\log_q n$.

VII. MAP Decoding of Nonbinary Hamming Codes

The parity-check matrix of p-ary Hamming code of length $n = (p^m-1)/(p-1)$ is an $p^m \times n$ matrix, whose columns are all nonzero, and nonproportional m-tuples over $\mathbb{F}_p$. If we add the overall parity check, then we obtain the extended p-ary Hamming codes $\mathcal{H}^p_m$.

The code $\mathcal{H}^p_m$ can be defined as a shortened $\mathcal{RM}_p(1,m)^\perp$ code. In other words, the set of coordinates of $\mathcal{H}^p_m$ is a subset, say $\mathcal{S}$, $|\mathcal{S}| = (p^m-1)/(p-1)+1$, of coordinates of $\mathcal{RM}_p(1,m)^\perp$. Hence, one can decode $\mathcal{H}^p_m$ as the code $\mathcal{RM}_p(1,m)^\perp$ assuming that on positions $\{0,1,\ldots,p^m-1\}\setminus\mathcal{S}$ erasures were received. Let $\arg(a)$, $a\in C$, be the function of extracting the argument of a complex number, i.e., if $a = |a|e^{\imath b}$ then $\arg(a) = b$.

The algorithm of MAP decoding of $\mathcal{H}_p(1,m)$ starts with computing values

$$\rho_j^{(\alpha)} = \begin{cases} \sum_{\beta\in\mathbb{F}_p} \omega^{-\alpha\beta}\Pr(y_j|\beta), & \text{if } j\in\mathcal{S} \\ 0, & \text{if } j\notin\mathcal{S} \end{cases}$$
$$\alpha\in\mathbb{F}_p, \qquad j = 0,\ldots,p^m-1,$$

their absolute values $t_j^{(\alpha)} = \ln|\rho_j^{(\alpha)}|$, and their arguments

$$a_j^{(\alpha)} = \arg\left(\rho_j^{(\alpha)}\right), \qquad \text{for } j = 0,\ldots,p^m-1.$$

Next we form vectors $\underline{\boldsymbol{t}}_j = \sum_{\alpha\in\mathbb{F}_p} t_j^{(\alpha)}x^\alpha$ and $\underline{\boldsymbol{a}}_j = \sum_{\alpha\in\mathbb{F}_p} a_j^{(\alpha)}x^\alpha$, and compute

$$\underline{\boldsymbol{s}} = (\boldsymbol{s}_0,\ldots,\boldsymbol{s}_{p^m-1}) = \mathbf{FHT}_{\mathbb{F}_p[\mathbb{R}]}(\boldsymbol{t}_0,\ldots,\boldsymbol{t}_{p^m-1})$$

and

$$\underline{\boldsymbol{b}} = (\boldsymbol{b}_0,\ldots,\boldsymbol{b}_{p^m-1}) = \mathbf{FHT}_{\mathbb{F}_p[\mathbb{R}]}(\boldsymbol{a}_0,\ldots,\boldsymbol{a}_{p^m-1}).$$

Using arguments similar to the ones used in Sections IV and V, we conclude that

$$h_j = e^{s_j^{(\alpha)}}e^{\imath b_j^{(\alpha)}} = \prod_{j=0}^{n}\rho_j^{(c'_{ij})}, \qquad i = 0,\ldots,p^m-1.$$

Next, we compute

$$\underline{\boldsymbol{g}} = (\boldsymbol{g}_0,\ldots,\boldsymbol{g}_{q^m-1}) = \mathbf{FHT}_{\mathbb{F}_p[\mathbb{R}]}(\boldsymbol{h}_0,\ldots,\boldsymbol{h}_{p^m-1}).$$

The entries of the vector $\underline{\boldsymbol{g}}$ are equal to

$$g_j^{(\alpha)} = \sum_{\underline{c}\in C_j^{(\alpha)}}\prod_{l=0}^{n}\rho_l^{c_l}.$$

Let us compute

$$s_j^{(\alpha)} = \sum_{\beta\in\mathbb{F}_p}\frac{g_j^{(\beta)}}{\rho_j^{(\beta)}}\rho_j^{(\beta\ominus\alpha)}$$

and

$$w_j^{(\alpha)} = \sum_{\beta\in\mathbb{F}_p}\omega^{\ominus\alpha\beta}s_j^{(\beta)}.$$

Now with the help of (5) and after a simple analysis, we obtain

$$z_j^{(\alpha)} = \frac{w_j^{(\alpha)}}{pg_j^{(\alpha)}} = \Pr(c_j = \alpha|\underline{y}).$$

This finishes the algorithm. It is easy to see that the complexity of this algorithm is the same as the complexity of MAP decoding of the nonbinary RM-1 codes, i.e., $pn\log_p n$.

References

[1] A. Ashikhmin and S. Litsyn, "Quasi-optimal biorthogonal-code decoding algorithms" (in Russian), *Radioelectronica*, vol. 31, no. 11, pp. 30–34, 1988. English translation in *Radio Electron. Commun. Syst.*, vol. 31, no. 11, pp. 26–30, 1988.

[2] ——, "A list algorithm for locating the maximal element in a Walsh spectrum" (in Russian), *Radioelectronica*, vol. 32, no. 3, pp. 15–22, 1990. English translation in *Radio Electron. Commun. Syst.*, vol. 32, 11, pp. 37–41, 1990.

[3] ——, "Fast decoding of first order Reed-Muller and related codes," *Des., Codes Cryptogr.*, vol. 7, pp. 187–215, 1996.

[4] ——, "Fast decoding of nonbinary orthogonal codes," *Applic. Algebra in Eng., Commun. Comput.*, vol. 7, no. 4, pp. 299–308, 1996.

[5] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284–287, Mar. 1974.

[6] Y. Be'ery and J. Snyders, "Optimal soft decision block decoders based on fast hadamard transform," *IEEE Trans. Inform. Theory*, vol. IT-32, pp. 355–364, May 1986.

[7] P. Delsarte, J.-M. Goethals, and F. J. MacWilliams, "On generalized Reed-Muller codes and their relatives," *Inform. Contr.*, vol. 16, pp. 403–442, 1974.

[8] M. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Trans. Inform. Theory*, vol. 41, pp. 1379–1396, Sept. 1995.

[9] C. R. P. Hartman and L. D. Rudolph, "An optimum symbol-by-symbol decoding rule for linear codes," *IEEE Trans. Infrm. Theory*, vol. IT-22, pp. 514–517, Sept. 1976.

[10] D. Gazelle and J. Snyders, "Reliability-based code-search algorithms for maximum-likelihood decoding of block codes," *IEEE Trans. Inform. Theory*, vol. 43, pp. 239–249, Jan. 1997.

[11] R. R. Green, "A serial orthogonal decoder," *JPL Space Programs Summary*, vol. 37–39-IV, pp. 247–253, 1966.

[12] I. J. Good, "The interaction algorithm and practical Fourier analysis," *J. Roy. Statist. Soc. (London)*, vol. B-20, pp. 361–372, 1958.

[13] I. Grushko, "Majority-logic decoding of generalized Reed-Muller codes" (in Russian), *Probl. Pered. Inform.*, vol. 26, no. 3, pp. 12–21, 1990. English translation in *Probl. Inform. Transm.*, vol. 26, pp. 189–196, 1990.

[14] S. A. Hirst, B. Honary, and G. Markarian, "Fast chase algorithm with an application in turbo decoding," *IEEE Trans. Commun.*, vol. 49, pp. 1693–1699, Oct. 2001.

[15] T. Kasami, S. Lin, and W. Peterson, "New generalizations of the Reed-Muller codes, Part 1: Primitive codes," *IEEE Trans. Inform. Theory*, vol. IT-14, pp. 189–199, Mar. 1968.

[16] T. Kasami, T. Takata, T. Fujiwara, and S. Lin, "On the optimum bit orders with respect to the state complexity of trellis diagrams for binary linear codes," *IEEE Trans. Inform. Theory*, vol. 39, pp. 242–245, Jan. 1993.

[17] ——, "On complexity of trellis structure of linear block codes," *IEEE Trans. Inform. Theory*, vol. 39, pp. 1057–1064, Jan. 1993.

[18] S. Litsyn, G. Mikhailovskaya, E. Nemirovsky, and O. Shekhovtsov, "Fast decoding of first order Reed-Muller codes in the Gaussian channel," *Probl. Contr. Inform. Theory*, vol. 14, no. 3, pp. 189–201, 1985.

[19] M. Meiler, J. Hagenauer, A. Schmidbauer, and R. Herzog, "Iterative decoding in the uplink of the CDMA-system IS-95(A)," in *Proc. Int. Symp. Turbo Codes*, Brest, France, 1997, pp. 208–211.

[20] A. M. Michelson and D. F. Freeman, "Viterbi decoding of the (63, 57) Hamming code—Implementation and performance results," *IEEE Trans. Commun.*, vol. 43, pp. 2653–2656, Nov. 1995.

[21] H. Miyakawa and T. Kaneko, "Decoding algorithm of error-correcting codes by use of analog weights," *Electron. Commun. Japan*, vol. 58, no. 1, pp. 18–27, 1975.

[22] D. E. Muller, "Application of boolean algebra to switching circuit design and to error detection," *IEEE Trans. Comput.*, vol. C–3, pp. 6–12, 1954.

[23] R. Pyndiah, "Near optimum decoding of product codes: Block turbo codes," *IEEE Trans. Commun.*, vol. 46, pp. 1003–1010, Aug. 1998.

[24] I. S. Reed, "A class of multiple-error-correcting codes and the decoding scheme," *IEEE Trans. Inform. Theory*, vol. IT-4, pp. 38–49, Sept. 1954.

[25] J. Snyders, "Reduced lists of error patterns for maximum likelihood soft decoding," *IEEE Trans. Inform. Theory*, vol. 37, pp. 1194–1200, July 1991.

[26] C.-E. Sundberg, "Asymptotically optimal soft decision decoding algorithms for Hamming codes," *Electron. Lett.*, vol. 13, no. 2, pp. 38–40, 1977.

[27] X. Wang and S. Wicker, "The BCJR trellis and trellis decoders for some BCH codes," *Appl. Algebra Eng. Commun. Comput.*, vol. 9, no. 6, pp. 521–546, 1999.

New Class of Nonlinear Systematic Error Detecting Codes

Mark Karpovsky, *Fellow, IEEE,* and
Alexander Taubin, *Senior Member, IEEE*

***Abstract*—A code C detects error e with probability $1-Q(e)$, if $Q(e)$ is a fraction of codewords y such that $y, y+e \in C$. We present a class of optimal nonlinear q-ary systematic (n, q^k)-codes (robust codes) minimizing over all (n, q^k)-codes the maximum of $Q(e)$ for nonzero e. We also show that any linear (n, q^k)-code V with $n \leq 2k$ can be modified into a nonlinear (n, q^k)-code C_V with simple encoding and decoding procedures, such that the set $E = \{e \,|Q(e) = 1\}$ of undetected errors for C_V is a $(k-r)$-dimensional subspace of V ($|E| = q^{k-r}$ instead of q^k for V). For the remaining $q^n - q^{k-r}$ nonzero errors, $Q(e) \leq q^{-r}$ for $q \geq 3$ and $Q(e) \leq 2^{-r+1}$ for $q = 2$.**

***Index Terms*—Minimax criterion for error detection, nonlinear systematic error detecting codes, robust error detection.**

I. Introduction

We present a construction of systematic error-detecting codes for the case when the distribution of errors in the channel is not known or difficult to model. A minimax criterion such that an error-detection capability for a code is optimized under the worst case scenario is the strategy taken for designing these codes.

Let $C \subseteq \mathrm{GF}(q^n)$ be an (n, q^k)-code and $e \in \mathrm{GF}(q^n)$. We define the error-detection probability given error e ($e \neq 0$) for the code C as

$$1 - Q(e) = 1 - q^{-k} \left| \{ y \,| y, y+e \in C \} \right|. \tag{1}$$

(We assume that all the codewords are equiprobable.)

The following lower bounds for error-masking probability $Q(e)$ have been proven in [1] for any q-ary code C of length n and any error e

$$Q(e) \geq |C|^{-1} \left[(q^n - 1)^{-1} |C| \left(|C| - 1 \right) \right], \qquad \text{for } q \geq 3 \tag{2}$$

and

$$Q(e) \geq 2 \, |C|^{-1} \left[0.5 \, |C| \left(|C| - 1 \right) (2^n - 1)^{-1} \right], \qquad \text{for } q = 2. \tag{3}$$

For a given block size n and number of information digits k our goal is to construct a systematic code such that the maximum of $Q(e)$ over all $e \neq 0$ is minimum. The problem can be formulated as follows. Formally, our criterion is $\min_C \max_{e \neq 0} Q(e)$, where the minimum is over all q-ary systematic (n, q^k)-codes.

We note that a similar minimax criterion has been used in the design of match filters to combat jamming and other modeling uncertainties for communication channels [5]. A similar problem was also considered in design of error detection mechanisms for computation channels (very large scale integration (VLSI) chips), when the distributions of errors (errors are manifestations of faults at the outputs of the chips) are difficult to characterize [3]. In the area of computer hardware testing, the minimax approach for optimal compression of test responses was described in [2].

In the following, we present solutions to the problem defined above for the cases $n = 2k$ and $n = 2k-1$ for $q = 2$, and $n = 2k$ for $q \geq 3$. We note that asymptotically optimal codes of length $n = 2m$ containing $q^{2m-1} - q^{m-1}$ codewords and detecting any error with probability at

Manuscript received July 9, 2002; revised December 19, 2003.

The authors are with the Reliable Computing Laboratory, Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215 USA (e-mail: markkar@bu.edu; taubin@bu.edu).

Communicated by S. Litsyn, Associate Editor for Coding Theory.

Digital Object Identifier 10.1109/TIT.2004.831844

0018-9448/04$20.00 © 2004 IEEE