國 立 交 通 大 學

電信工程學系

碩 士 論 文

RS 編碼和迴旋碼: 串接碼的軟式決定解碼法

Soft-Decision Decoding of Concatenated Reed-Solomon and Convolutional Codes

研 究 生：楊哲雄

指導教授：蘇育德 博士

西元 2006 年 8 月

RS 編碼和迴旋碼: 串接碼的軟式決定解碼法

Soft-Decision Decoding of Concatenated Reed-Solomon and Convolutional Codes

研 究 生： 楊哲雄　　Student: Che-Hsiung Yang

指導教授： 蘇育德 博士　　Advisor: Dr. Yu T. Su

國立交通大學

電信工程學系碩士班

碩士論文

A Thesis Submitted to

Department of Communication Engineering

College of Electrical and Computer Engineering

National Chiao Tung University

In Partial Fulfillment of the

Requirments for the Degree of

Master of Science

In Communication Engineering

Hsinchu, Taiwan, Republic of China

August 2006

誌　謝

首先感謝指導老師蘇育德教授兩年來的指導， 使得論文能更順利完成。 在這兩年的時間裡面， 老師的諄諄教誨讓我在通訊領域上有了更加深入的了解，也讓我了解到了一些做人處世的道理。 另外也要感謝口試委員林茂昭教授，呂忠津教授， 楊谷章教授以及吳文榕教授給予的寶貴意見，以彌補這份論文上的缺失跟不足之處。 也要感謝實驗室的學長姐、同學及學弟妹的幫忙還有鼓勵，讓我不僅在學習的過程中獲益匪淺，同時也為這兩年的生活增添了許多歡樂。

最後，我也要感謝一直關心我、鼓勵我的家人，沒有他們的支持我沒辦法這麼順利的完成論文，謹獻上此論文，代表我最深的謝意！

RS 編碼和迴旋碼: 串接碼的軟式決定解碼法

研究生: 楊哲雄　　指導教授: 蘇育德 博士

國立交通大學電信工程學系碩士班

中文摘要

Reed-Solomon (RS)-迴旋串接碼是一種非常有效的錯誤更碼。由於其強大改錯能力，這種碼已經被應用在很多的通訊和儲存系統。眾所皆知，串接碼的重複解法相對於傳統的串接碼在性能上有明顯的改善。但是，除了 Jiang 及 Narayanan（JN）兩氏在兩年前提出利用信心傳遞法推導出來的複雜度頗高的方法之外由於缺乏有效的軟式輸出演算法，幾乎所有的 RS 碼或 RS-迴旋串接碼之解碼器都用硬式解碼法。

很多快速限定距離軟式 RS 解碼演算法在 1990 年代初期即被提出，但其解碼能力與硬式解碼法比較改善有限。1997 年 Sudan 是第一位提出可以對低編碼率 RS 碼超越其硬式解碼設計距離的有效軟式解碼法的人。兩年之後他和他的學生 Guruswami 更進一步提出簡稱爲 GS 演算法的改良版本。這個新演算法由兩個部分所組成: 內插和因式分解。Kötter 與 Vardy（KV）接著又提出另一軟式解碼演算法。他們藉由轉換通道可靠度到內差點的點數，大大減低了 GS 演算法的解碼複雜度。由於 KV 演算法的複雜度仍然高，Kötter 與其他的人再藉由轉換接收的值可以改善重複的次數，

使得改良後的 KV 演算法變得可行。

本篇論文的目的在於探討 RS 碼軟式輸入和軟式輸出 (SISO)解碼的可能性，希望讓 RS-迴旋串接碼的重複解變爲實際可能。我們評估了不同軟式 RS 解碼法的效能，檢視它們對單獨 RS 碼及串接碼所能提供的改善幅度。我們也提出一種綜合改良式 KV 演算法和 JN 軟式演算法的遞迴式 SISO RS-迴旋串接碼解碼法並透過電腦模擬估算其解碼能力。

Soft-Decision Decoding of Concatenated Reed-Solomon and Convolutional Codes

Student : Che-Hsiung Yang Advisor : Yu T. Su

Department of Communications Engineering

National Chiao Tung University

Abstract

Serially concatenated Reed-Solomon (RS) and convolutional codes form a class of powerful error-correcting codes that has been used in many applications. It is known that iterative decoding of concatenated codes provides significant performance gain with respect to conventional concatenated decoding. However, almost all RS decoders were based on the hard-decision decoding (HDD) premise due to the lack of an efficient soft-output decoding algorithm for RS codes.

Several fast bounded-distance SDD algorithms were proposed in the early 90's. The Guruswami and Sudan (GS) algorithm, which consists of two decoding phases– interpolation and factorization, is the first efficient SDD algorithm for RS codes that is capable of correcting beyond the designed (HD) distance. The Koetter-Vardy (KV) soft-decision algorithm reduces the decoding complexity of the GS algorithm by converting the probabilistic reliability information into a set of interpolation points along with their multiplicities. As the complexity of KV algorithm is still relatively high, a improvement that uses a transformation of the received word to reduce the number of iterations was then presented, making the implementation of the KV algorithm feasible.

The purpose of this thesis is to investigate the feasibility of soft-input soft-output (SISO) RS decoders so that iterative decoding of concatenated RS-convolutional codes

becomes possible. We present a SISO RS decoder that is based on the modified Koetter-Vardy (KV) and Jiang-Narayanan soft-decision algorithms.

Contents

List of Figures

Chapter 1

Introduction

Reed-Solomon (RS) codes are maximum distance separable (MDS) codes which provide powerful error correction capability with minimum number of overhead symbols. A classical hard-decision (HD) decoder for an (n, k) RS code can correct up to $t = \lfloor d_{min}/2 \rfloor$ errors where $d_{min} = (n-k+1)$ is the minimum distance of the code. Furthermore, as RS codewords consist of non-binary symbols the correction of a single symbol results in the correction of more than one of the constituent bits, they are well suited to the correction of burst errors. This fact had motivated Forney to propose a serial concatenated coding scheme with a RS outer code and a convolutional inner code [11]. For these reasons RS codes have found wide applications in both digital communication and storage systems. The ubiquitous nature of this class of codes continues to fuel research into the associated decoding algorithm even almost fifty years after their introduction.

Most concatenated RS-convolutional coding system use HD decoding (HDD) algorithm because soft-decision decoding (SDD) algorithms usually demand very high computational effort. Traditional HD RS decoding algorithms are efficient because they are algebraic; that is, they exploit the underlying algebraic structure of the code to generate a system of equations that is solved using finite field arithmetic. However, it is known that significant performance gain can be achieved by using a SD decoder for RS codes. Unfortunately, an algebraic decoder based on finite field arithmetic does not appear to be compatible with the real-valued, soft information available either from the channel

or from the convolutional decoder output. Therefore, it has been a research challenge to develop an SD RS decoder.

An early example of SDD of block codes is given by Wagner decoding and its generalizations. The reliability-based proposals of Forney [12] and Chase [13] have attracted many followers. Using the binary image expansions of M-ary symbols, Vardy and Be'ery [15] showed that RS codes can be decomposed into BCH subfield subcodes which are glued together using glue vectors. Even though this decomposition significantly reduces the trellis complexity of maximum likelihood (ML) decoding of RS codes, the complexity still grows exponentially with the code length and d_{min} and it is thus infeasible for decoding long codes. Ponnampalam and Vucetic [16] suggest an SDD algorithm to reduces the complexity to generate soft output efficiently. Similarly, one can also use reliability based ordered statistics decoding (OSD) [17] and its variations [18] for soft decision decoding of RS codes. Related works include the hybrid algorithm by Hu and Lin [19] and the box and match algorithm (BMA) [20] by Fossorier and Valembois. OSD based algorithms are quite efficient for practical RS codes even though they do not take the structure of the RS codes into account.

In 1997 Madhu Sudan [1] presented a polynomial-time algorithm for decoding certain low-rate RS codes beyond the classical $d/2$ error-correcting bound. Two years later, he and his student, Guruswami discovered [2] a significantly improved version of Sudan's algorithm, which was capable of decoding virtually every RS code at least somewhat, and often significantly, beyond the $d/2$ limit. Several subsequent investigations were able to find low-complexity realizations for the key steps in the Guruswami-Sudan (GS) algorithm, thus making GS a genuinely practical engineering alternative. In particular, Köetter and Vardy (KV) [9] have proposed an algebraic SDD algorithm by extending the list GS decoder to include a method for converting soft information into algebraic conditions. The KV SDD procedure shows a lot of promise from the point of view of error correcting performance. However, the algorithm is still quite computationally complex

and not straightforward to implement in VLSI. K0̈tter *et al.* [7, 8] later introduce techniques that reduce the complexity of interpolation-based decoders to the point where an efficient VLSI implementation is feasible. The purpose of this thesis is to evaluate the SDD performance of concatenated RS-convolutional coding systems. We also suggest an iterative decoder structure and show that the corresponding performance does improve as the number of iterations increases.

The rest of this thesis is organized as follows. In Chapter 2, we will prove the GS error correction capability beyond the classical error correction bound. In this chapter, we also present the realizable interpolation and factorization algorithm that makes the GS algorithm a genuinely practical engineering alternative in storage and transmission systems. In Chapter 3, we show how the channel reliability can be translated into the multiplicities that makes the HD GS algorithm into the SD KV algorithm. In Chapter 4, we describe a algorithm that reduces the complexity and memory requirements of interpolation-based decoders. A modified factorization algorithm is presented in this chapter as well. In Chapter 5, we present a soft-input soft-output (SISO) iterative decoding algorithm for serial concatenated convolutional and RS coding systems. Chapter 6 provides numerical examples associated with various SDD algorithms discussed in the earlier chapters.

Chapter 2

The Guruswami-Sudan Decoding Algorithm for RS Codes

A polynomial-time algorithm for decoding certain low-rate RS codes beyond the classical $d/2$ error-correcting bound was invented by Sudan [1] who formulated the decoding problem as follows.

- **Input**: A field F; n distinct pairs of elements $\{x_i, y_i\}_{i=1}^{n}$ from $F \times F$; and integers d and t.
- **Output**: A list of all functions $f : F \to F$ satisfying: $f(x)$ is a polynomial in x of degree at most d with $|\{i|f(x_i) = y_i\}| \geq t$.

This problem is also known as the **list decoding** problem. A much improved version of Sudan's algorithm was discovered by Guruswami and Sudan (GS) [2].

2.1 A First Look at the GS Algorithm

This section provides an overview of the GS algorithm. We give a motivating example, an informal description of the algorithm and several numerical examples.

Let $(\alpha_1, \cdots, \alpha_n)$ be a fixed list of n distinct elements of F =GF(q), called the support set of the code. The encoding process of an (n, k) RS code is that of mapping a vector $(f_0, f_1, \cdots, f_{k-1})$ of k information symbols into an n-symbol codeword $(x_1, \cdots, x_n)$ by

polynomial evaluation, i.e.,

$$(x_1, \cdots, x_n) = (f(\alpha_1), \cdots, f(\alpha_n)) \tag{2.1}$$

where

$$f(x) = f_0 + f_1 x + \cdots + f_{k-1} x^{k-1} \tag{2.2}$$

The corresponding RS code consists of all n-vectors of the form in (2.1), where $f(x)$ is a polynomial of degree $< k$.

It is well-known that this code has minimum Hamming distance $d = n - k + 1$ and, therefore, is capable of correcting up to

$$t_0 = \left\lfloor \frac{n-k}{2} \right\rfloor \tag{2.3}$$

errors. Conceptually, this may be accomplished as follows. The decoder searches the Hamming sphere of radius t_0 centered at the received word for codewords. If the sphere contains a unique codeword, it is the decoder's output. Otherwise, the decoder reports a failure. (This strategy is called *bounded distance* decoding (BDD).) The decoding sphere cannot contain more than one codeword, since the minimum distance of the code is $> 2t_0$. If we attempt to correct more than t_0 errors by increasing the decoding radius, it is possible for the decoding sphere to contain more than one codeword, in which case the decoder will fail. For this reason, conventional wisdom asserts that the code is not capable of correcting more than t_0 errors. Nevertheless, if we examine the *probability* that the decoding sphere will contain multiple codewords, rather than the *possibility*, we may reach a different conclusion.

Example 1 *Consider the* $(32, 8)$ *RS code over* $GF(32)$*, with* $d = 25$ *and* $t_0 = 12$*. If the decoding radius is taken to be* $t = 13$*, and the transmitted codeword suffers 13 errors, it is possible for the decoding sphere to contain two codewords: the transmitted codeword (which we will call the causal codeword) and one other, a noncausal codeword at distance*

12 or 13 from the received word. However, it can be shown that the probability of this unfavorable happening is 2.08437×10^{-12}*! In short, the code is capable of correcting virtually all patterns of 13 errors, despite having a conventional error-correcting capability of only 12.*

The above example suggests that it might be possible to design a decoding algorithm for RS codes capable of correcting more than t_0 errors. The GS list decoding algorithm does just this. It is a polynomial-time (Conservatively, the time complexity is $O(n^2m^4)$, where n is the code length and m is the interpolation multiplicity). algorithm for correcting up to t_{GS} errors, where t_{GS} is the largest integer strictly less than $n - \sqrt{(k-1)n}$, i.e.,

$$t_{GS} = n - 1 - \lfloor \sqrt{(k-1)n} \rfloor \tag{2.4}$$

It is easy to show that $t_{GS} \geq t_0$, and often t_{GS} is considerably greater than t_0 (see the examples below). Asymptotically, for *RS* codes of rate R, the conventional decoding algorithms will correct a fraction $\tau_0 = (1-R)/2$ of errors, while the GS algorithm can correct up to $\tau_{GS} = 1 - \sqrt{R}$. Figure 2.1 shows this fact. The GS decoder has an adjustable integer parameter $m \geq 1$ called the interpolation multiplicity. Associated with the interpolation multiplicity m is positive integer $t = t_m$, called the designed decoding radius. Given a received word, the $GS(m)$ decoder returns a list that includes all codewords with Hamming distance t_m or less from the received word, and perhaps a few others. The exact formula for t_m is a bit complicated, but for now it suffices to say that

$$t_0 \leq t_1 \leq t_2 \leq \cdots$$

and there exists an integer m_0 such that

$$t_{m0} = t_{m_0+1} = \cdots = t_{GS}$$

Here is an overview of the $GS(m)$ algorithm (a detailed description will be given in Section 2.5). Suppose $C = (f(\alpha_1), \cdots, f(\alpha_n))$ is the transmitted codeword, where $f(x)$

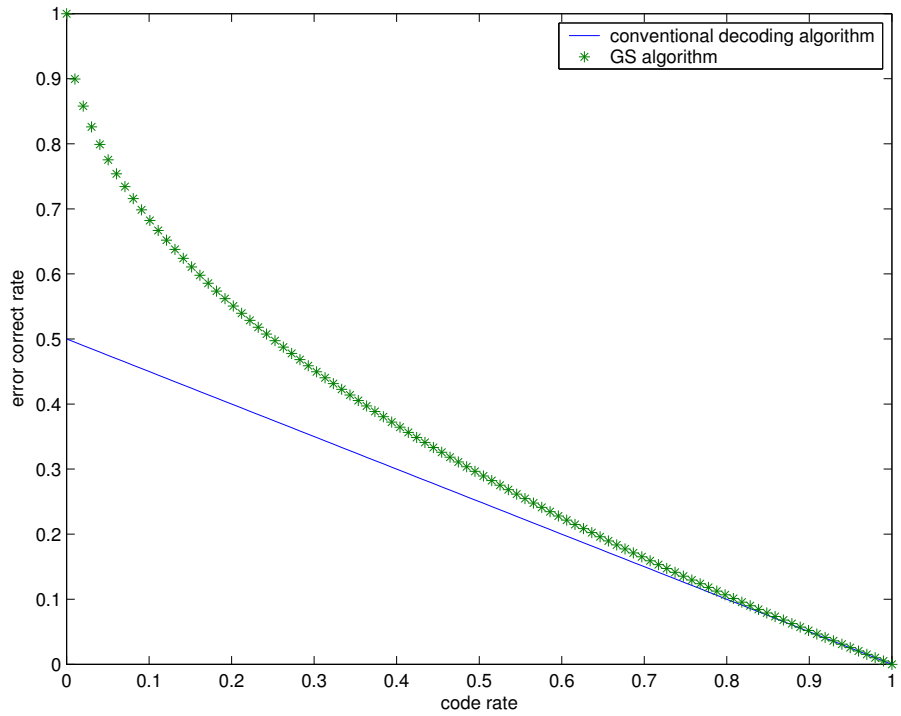


Figure 2.1: Error-correcting capacity plotted against the rate of the code

is a polynomial of degree $< k$, and that C is received as $R = (\beta_1, \cdots, \beta_n)$. Let $p(x)$ be any polynomial of degree $< k$ that maps to an RS codeword with Hamming distance $\leq t_m$ from R, i.e.,

$$|\{i : p(\alpha_i) \neq \beta_i\}| \leq t_m$$

The $GS(m)$ decoder "*finds*" $p(x)$ as follows.

1. The interpolation step. Given the received vector $R = (\beta_1, \cdots, \beta_n)$, the decoder constructs a two-variable polynomial

$$Q(x, y) = \sum_{i,j} a_{i,j} x^i y^j$$

with the property that Q has a zero of multiplicity m (exact definition is given in Section 2.3) at each of the points (α_i, β_i), and for which the $(1, k-1)$ weighted degree (exact definition is given in Section 2.2) of $Q(x, y)$ is as small as possible.

2. The factorization step. The decoder then finds all factors of $Q(x, y)$ of the form

$y - p(x)$, where $p(x)$ is a polynomial of degree $k-1$ or less. Let

$$\mathcal{L} = \{p_1(x), \cdots, p_L(x)\}$$

be the list of polynomials produced by this step. The polynomials (codewords) $p(x) \in \mathcal{L}$ are of three possible types:

(a) *Type 1.* The transmitted, or causal, codeword.

(b) *Type 2.* Codewords with Hamming distance $\leq t_m$ from R, which we call plausible codewords.

(c) *Type 3.* Codewords with distance $> t_m$ from R, which we call implausible codewords.

In Section 2.5, we give a proof of the following theorem.

Theorem 1 *If the $GS(m)$ decoding algorithm is used, all plausible codewords will be in $\mathcal{L}$. In particular, the transmitted codeword will be in $\mathcal{L}$ if the number of channel errors is $\leq tm$. The list may also contain implausible codewords, but the total number of codewords in the list, plausible and implausible, will satisfy $L \leq L_m$, where the exact determination of L_m is given in (2.31) of Section 2.5, but which is conservatively estimated by*

$$L_m < (m + \frac{1}{2})\sqrt{\frac{n}{k-1}} \tag{2.5}$$

Example 2 *Consider again the $(32, 8)$ RS code over $GF(32)$, with $r = 24$ and $d = 25$. Its conventional error-correcting capability is $t_0 = 12$ errors, but by (2.4), the GS algorithm can correct up to $t_{GS} = 17$ errors! The value of the designed decoding radius t_m as a function of the interpolation multiplicity m is given in the table below, together with the exact value of L_m as given in (2.31). The value*

$$\overline{L}(t) = q^{-r} \sum_{s=0}^{t} \binom{n}{s} (q-1)^s$$

which is the average number of codewords in a randomly chosen sphere of radius t, gives a heuristic upper bound on the probability that the decoding sphere will contain a noncausal codeword. Values of m that do not afford a larger value of t_m than the previous value are omitted. For example, in the present example, $t_2 = t_3 = 15$, and so $m = 3$ is omitted from the table. Similarly, $t_5 = t_6 = \cdots = t_{119} = 16$: It is interesting to note the growth

m	t_m	L_m	$\overline{L}(t_m)$
classical	12	1	1.36305×10^{-10}
1	14	2	2.74982×10^{-07}
2	15	4	0.0000102619
4	16	8	0.000339205
$m_0 = 120$	17	256	0.00993659

in the required value of m as t increases from 16 ($m = 4$) to 17 ($m = 120$), which indicates that $t = 16$ is the practical limit for the GS algorithm in this case.

Example 3 *Similarly, for the (16,4) RS code over GF(16) ($t_0 = 6$ and $t_{GS} = 9$), we have Here we see that for $t = t_{GS} = 9$ the interpolation multiplicity may be prohibitively*

m	t_m	L_m	$\overline{L}(t_m)$
classical	6	1	0.000336183
1	7	2	0.00728043
2	8	4	0.124465
$m_0 = 28$	9	120	1.68692

large, so that $t = 8$ is the practical limit.

Example 4 *For the (6,4) RS code over GF(7), $t_0 = t_{GS} = 1$ and This is a rare example*

m	t_m	L_m	$\overline{L}(t_m)$
classical	1	1	0.7551

where the GS algorithm provides no improvement over conventional decoding.

m	t_m	L_m	$\overline{L}(t_m)$
classical	16	1	2.609×10^{-14}
$m_0 = 112$	17	120	9.35×10^{-11}

Example 5 *For the (255,223) RS code over GF*(256) *(*$t_0 = 16$ *and* $t_{GS} = 17$*): Not until* $m = 112$ *does the GS algorithm offer an improvement over conventional decoders, and even then the improvement is only one extra error corrected. With the decoding complexity* $O(m^4)$*, it seems pointless to try to correct the extra error.*

Example 6 *The (255,239) RS code over GF*(256) *(*$t_0 = 8$ *and* $t_{GS} = 8$*):*

m	t_m	L_m	$\overline{L}(t_m)$
classical	8	1	2.0853×10^{-05}

2.2 Polynomials in Two Variables I: Monomial Orders and Generalized Degree

In this section, we present a self-contained introduction to the algebraic fundamentals of two-variable polynomials. These fundamentals include weighted monomial orderings, generalized degree functions, and certain related combinatorial results. It is the basis to prove the algorithm of $GS(m)$ algorithm.

If F is a field, we denote by $F[x, y]$ the ring of polynomials in x and y with coefficients from F. A polynomial $Q(x, y) \in F[x, y]$ is, by definition, a finite sum of monomials, viz.,

$$Q(x, y) = \sum_{i.j \geq 0} a_{i,j} x^i y^j \tag{2.6}$$

where only a finite number of the coefficients $a_{i,j}$ are nonzero. The summation in Eq. (2.6) is two-dimensional, but often it is desirable to have a one-dimensional representation instead. To do this, we need to have a linear ordering of the set of monomials

$$\mathrm{M}[x, y] = \{x^i y^j : i, j \geq 0\}$$

There are many possible monomial orderings, but for us the most important ones are the weighted degree (WD) monomial orders. A WD monomial order is characterized by a pair $w = (u, v)$ of nonnegative integers, not both zero. For a fixed w, the w-degree of the monomial $x^i y^j$ is defined as

$$\deg_w x^i y^j = ui + vj$$

Definition 1 *The w-lex order is defined as follows:*

$$x_1^i y_1^j < x_2^i y_2^j$$

if either $ui_1 + vj_1 < ui_2 + vj_2$, or $ui_1 + vj_1 = ui_2 + vj_2$ and $i_1 < i_2$. w-revlex order is similar, except that the rule for breaking ties is $i_1 > i_2$. (In the special case $w =$ (1, 1), these orderings are called graded-lex, or grlex, and reverse graded-lex, or grevlex, respectively.)

The w-revlex order plays an important role in our discourse.

Example 7 *For any monomial order, we have $xy < x^2y$. Also, $xy^2 <_{grlex} x^2y$, but $x^2y <_{grevlex} xy^2$. Finally, if $w =$ (1, 3), $x^6 <_{wrevlex} x^3y <_{wrevlex} y^2$.*

Let "$<$" be a fixed monomial ordering:

$$1 = \phi_0(x, y) < \phi_1(x, y) < \phi_2(x, y) < \cdots$$

With respect to this ordering, every nonzero polynomial in $F[x, y]$ can be expressed uniquely in the form

$$Q(x, y) = \sum_{j=0}^{J} a_j \phi_j(x, y)$$

for suitable coefficients $a_j \in F$, with $a_J \neq 0$. The integer J is called the rank of $Q(x, y)$, and the monomial ϕ_J is called the leading monomial of $Q(x, y)$. We indicate this notationally by writing $\text{Rank}(Q) = J$ and $\text{LM}(Q) = \phi_J(x, y)$.

In the case of a WD order, the weighted degree of the leading monomial ϕ_j is also called the weighted degree, or w-degree, of $Q(x,y)$, denoted by $\deg_w Q$. Thus,

$$\deg_w Q(x,y) = \max \ \ \deg_w \phi(x,y) : a_j \neq 0\}$$

If $\phi_0(x,y) < \phi_1(x,y) < \cdots$ is a fixed monomial ordering, and $\phi = x^i y^j$ is a particular monomial, the index of ϕ, denoted by $\mathrm{Ind}(\phi)$, is defined as the unique integer K such that $\phi_K(x,y) = \phi$.

Example 8 *Here is a listing of the first few monomials, in the "natural" two-dimensional array, but labelled according to (1,3)-revlex order:*

i=	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
j=0	0	1	2	3	5	7	9	12	15	18	22	26	30	35	40	45	$\cdots$
j=1	4	6	8	10	13	16	19	23	27	31	36	41	46				
j=2	11	14	17	20	24	28	32	37	42	47							
j=3	21	25	29	33	38	43	48										
j=4	34	39	44														

Thus, we have ϕ_0= 1, $\phi_1 = x$, $\phi_2 = x^2$, $\phi_3 = x^3$, $\phi_4 = y$,$\cdots$, $\phi_{48} = x^6y^3$, $\cdots$. Also, $\mathrm{Ind}(xy) = 6$, $\mathrm{Ind}(x^2y^2) = 17$, $\mathrm{Ind}(x^9y^2) = 47$, etc. For $(1,v)$ revlex order, the numbers $\mathrm{Ind}(x^K)$ and $\mathrm{Ind}(y^L)$ are especially important, so we introduce a special notation for them.

$$A(K,v) \triangleq Ind(x^K) \tag{2.7}$$

$$B(L,v) \triangleq Ind(y^L) \tag{2.8}$$

It is understood that the underlying monomial order is $(1,v)$-revlex. In terms of the two-dimensional array given above, the numbers $A(K,v)$ appear in the $j = 0$ row and the numbers $B(L,v)$ appear in the i=0 column. Thus, with $v = 3$, we have

x	0	1	2	3	4	5	6	7	$\cdots$
A(x,3)	0	1	2	3	5	7	9	12	$\cdots$
B(x,3)	0	4	11	21	34	50	69	90	$\cdots$

We note that x^K is the first monomial of $(1, v)$-degree K, and y^L is the last monomial of $(1,\ v)$-degree vL, so that

$$A(K, v) = |\{(i, j) : i + vj < K\}| \tag{2.9}$$

$$B(L, v) = |\{(i, j) : i + vj \leq Lv\}| - 1 \tag{2.10}$$

We conclude this section with a consideration of two-variable polynomials of the form

$$Q(x, y) = \sum_{j=0}^{J} a_j \phi_j(x, y)$$

where $\phi_0 < \phi_1 < \cdots$ is $(1,v)$-revlex order, and $\{a_0, a_1, \cdots, a_J\}$ are arbitrary elements of F. (N.B., We do not assume that $a_J \neq 0$.)

Two important questions that will arise are (1) what is the $(1,\ v)$-degree of $Q(x, y)$ and (2) what is the y-degree, i.e., the (0,1)-degree, of $Q(x, y)$? We know that

$$\deg_{1,v} Q(x, y) \leq \max \deg_{1,v} \phi_j(x, y) : j = 0, \cdots, J\}$$

$$\deg_{0,1} Q(x, y) \leq \max \deg_{0,1} \phi_j(x, y) : j = 0, \cdots, J\}$$

Thus, if we define[1]

$$D(u, v; J) = \max \deg_{u,v} \phi_j(x, y) : j = 0, \cdots, J\}$$

we have the upper bounds

$$\deg_{1,v} Q(x, y) \leq D(1, v; J)$$

$$\deg_{0,1} Q(x, y) \leq D(0, 1; J)$$

A new definition is needed. Let $A = \{0 = a_0 < a_1 < a_2 < \cdots\}$ be an increasing sequence of integers, and let $x \geq 0$ be a nonnegative real number. The rank of apparition

[1]It is understood that the monomial order is $(1, v)$-revlex

of x with respect to A, denoted by $r_A(x)$, is the unique index K such that $a_K \leq x < a_{K+1}$. Alternatively,

$$r_A(x) = \max\{K : a_K \leq x\} \tag{2.11}$$

$$= \min\{L : x < a_{L+1}\} \tag{2.12}$$

Theorem 2 *Given a fixed v, define sequences $\{a_K = A(K, v)\}$ and $\{b_L = B(L, v)\}$. Then*

$$D(1, v : J) = r_A(J) \tag{2.13}$$

$$D(0, 1 : J) = r_B(J) \tag{2.14}$$

Corollary 1 *For $v \geq 1, K \geq 0$,*

$$\frac{K^2}{2v} < A(K, v) \leq \frac{(K + v/2)^2}{2v} \tag{2.15}$$

Corollary 2 *For $v \geq 1, J \geq 0$,*

$$\left\lfloor \sqrt{2vJ} - \frac{v}{2} \right\rfloor \leq r_A(J) \leq \left\lfloor \sqrt{2vJ} \right\rfloor - 1 \tag{2.16}$$

Corollary 3 *For $v \geq 1, J \geq 0$,*

$$\left\lfloor \sqrt{\frac{2J}{v}} - \frac{v+2}{2v} \right\rfloor \leq r_B(J) \leq \left\lfloor \sqrt{\frac{2J}{v}} \right\rfloor \tag{2.17}$$

Detailed proofs of the above theorem and corollaries can be found in [3].

2.3 Polynomials in Two Variables II: Zeros and Multiple Zeros

In this section, we continue with our study of bivariate polynomials and focus on the notion of a zero or a multiple zero of such polynomials.

If $Q(x, y) \in F[x, y]$, and $Q(\alpha, \beta) = 0$, we say that Q has a zero at (α, β) (Alternatively, we say that the curve $Q(x, y) = 0$ passes through the point (α, β)). We shall be interested in polynomials with multiple zeros.

Definition 2 *We say that* $Q(x,y) = \sum_{i,j} a_{i,j} x^i y^j \in F[x,y]$ *has a zero of multiplicity, or order* m *at* $(0,0)$*, and write*

$$ord(Q : 0,0) = m$$

if $Q(x,y)$ *involves no term of total degree less than* m*, i.e.,* $a_{i,j} = 0$ *if* $i + j < m$*. Similarly, we say that* $Q(x,y)$ *has a zero of order* m *at* (α,β) *and write*

$$ord(Q : \alpha,\beta) = m$$

if $Q(x+\alpha, y+\beta)$ *has a zero of order* m *at* $(0,0)$*.*

Example 9 *Let* $Q(x,y) = x^2y + xy^3 + x^3y$*. Then* Q *has a zero of multiplicity* 3 *(a "triple zero") at* $(0,0)$*. Similarly,* $P(x,y) = (x-\alpha)^2(y-\beta) + (x-\alpha)(y-\beta)^3 + (x-\alpha)^3(y-\beta)$ *has a triple zero at* (α,β)*.*

To calculate ord$(Q : \alpha,\beta)$, we need to be able to express $Q(x+\alpha, y+\beta)$ as a polynomial in x and y. The following theorems, due to H. Hasse, tell us one way to do this. We begin with the one-variable version of Hasse's theorem, because it serves as a simplified introduction to the two-variable case.

Theorem 3 *If* $Q(x) = \sum_i a_i x^i \in F[x]$*, then for any* $\alpha \in F$*, we have*

$$Q(x+\alpha) = \sum_r Q_r(\alpha) x^r \tag{2.18}$$

where

$$Q_r(x) = \sum_i \binom{i}{r} a_i x^{i-r} \tag{2.19}$$

which is called the r*th Hasse derivative of* $Q(x)$[2]*.*

Note that

$$Q_r(\alpha) = \text{Coeff}_{x^r} Q(x+\alpha) = \sum_i \binom{i}{r} a_i \alpha^{i-r} \tag{2.20}$$

[2]The alternative notation $D_r Q(x)$ instead of $Q_r(x)$ is sometimes used.

Furthermore, (2.16) is Taylor's formula (without remainder) when F has characteristic 0, since in that case,

$$Q_r(x) = \frac{1}{r!}\frac{d^r}{dx^r}Q(x) \tag{2.21}$$

We also have

Corollary 4

$$Q(x) = \sum_{r\geq 0} Q_r(\alpha)(x-\alpha)^r$$

Theorem 4 *Let $Q(x,y) = \sum_{i,j} a_{i,j}x^i y^j \in F[x,y]$. For any $(\alpha.\beta) \in F^2$, we have*

$$Q(x+\alpha, y+\beta) = \sum_{r,s} Q_{r,s}(\alpha,\beta)x^r y^s \tag{2.22}$$

where

$$Q_{r,s}(x,y) = \sum_{i,j} \binom{i}{r}\binom{j}{s} a_{i,j}x^{i-r}y^{j-s} \tag{2.23}$$

which is called the (r,s)th Hasse (mixed partial) derivative of $Q(x,y)$[3].

Note that (2.19) is Taylor's formula (without remainder) when F has characteristic 0, since in that case,

$$Q_{r,s}(x,y) = \frac{1}{r!s!}\frac{\partial^{r+s}}{\partial x^r \partial y^s}Q(x,y)$$

Note also the alternative, but equivalent, formula:

$$Q_{r,s}(\alpha,\beta) = \text{Coeff}_{x^r,y^s}Q(x+\alpha, y+\beta) \tag{2.24}$$

One can easily show

Corollary 5

$$Q(x,y) = \sum_{r,s} Q_{r,s}(\alpha,\beta)(x-\alpha)^r(y-\beta)^s$$

and

[3]Sometimes we use the alternative notation $D_{r,s}Q(x,y)$ instead of $Q_{r,s}(x,y)$.

Corollary 6 *The polynomial $Q(x,y)$ has a zero of order m at (α,β) if and only if*

$$Q_{r,s}(\alpha,\beta)=0 \text{ for all } r \text{ and } s \text{ such that } 0\le r+s<m$$

which follows directly from Corollary 5.

Corollary 7 *If $\widetilde{Q}(x,y)=xQ(x,y)$, then*

$$\widetilde{Q}_{r,s}(x,y)=Q_{r-1,s}(x,y)+xQ_{r,s}(x,y)$$

Similarly,if $\widetilde{Q}(x,y)=yQ(x,y)$, then

$$\widetilde{Q}_{r,s}(x,y)=Q_{r,s-1}(x,y)+yQ_{r,s}(x,y)$$

For proofs of these corollaries see [3].

2.4 The Interpolation and Factorization Theorems

In this section, we will state and prove the two basic theorems that support the GS algorithm. We call these theorems the Interpolation Theorem and the Factorization Theorem.

2.4.1 The Interpolation Theorem

Suppose a nonnegative integer $m(\alpha)$ is assigned to each element $\alpha\in F$, and we are asked to construct a polynomial $f(x)$ of least degree that has a zero of multiplicity $m(\alpha)$, at $x=\alpha$, for all $\alpha\in F$. Clearly a minimum-degree solution to this one-dimensional interpolation problems is

$$\begin{aligned} f(x) &= \prod_{\alpha\in F}(x-\alpha)^{m(\alpha)} \\ \deg f(x) &= \sum_{\alpha\in F} m(\alpha) \end{aligned}$$

We are interested in the analogous two-dimensional interpolation problem: Given a required multiplicity $m(\alpha,\beta)$ for each $(\alpha,\beta)\in F^2$, construct a low-degree polynomial

$Q(x,y)$ that has zeros of the required multiplicity. This is a much harder problem, in general, but the following theorem gives a useful upper bound on the minimum required degree.

Theorem 5 (**The Interpolation Theorem**) *Let $\{m(\alpha,\beta) : (\alpha,\beta) \in F^2\}$ be a multiplicity function as above and let $\phi_0 < \phi_1 < \cdots$ be an arbitrary monomial order. Then there exists a nonzero polynomial $Q(x,y)$ of the form*

$$Q(x,y) = \sum_{i=0}^{C} a_i \phi_i(x,y) \tag{2.25}$$

where

$$C = \sum_{\alpha,\beta} \binom{m(\alpha,\beta)+1}{2}$$

which has a zero of multiplicity m at (α,β), for all $(\alpha,\beta) \in F^2$.

Proof. By *Corollary 6*, $Q(x,y)$ has a zero of multiplicity m at (α,β) if and only if

$$Q_{r,s}(\alpha,\beta) = 0 \quad \text{for all } (r,s) \text{ such that } 0 \le r+s < m(\alpha,\beta) \tag{2.26}$$

There are $\binom{m(\alpha,\beta)+1}{2}$ choices for (r,s) in (2.24), and from (2.21), each such choice imposes one homogeneous linear constraint on the coefficients ai. In total there are C such linear constraints imposed on the $C+1$ coefficients $a_0, a_1, \cdots, a_C$. It follows that there must be at least one nonzero solution to this set of equations, which corresponds to a nonzero polynomial $Q(x,y)$ of the form in (2.23) with the required multiplicities. ■

Corollary 8 *For any (u,v), there is a nonzero polynomial $Q(x,y)$ with required zero multiplicities whose (u,v)-degree is strictly less than $\sqrt{2uvC}$.*

Proof: Take $\{\phi_j(x,y)\}$ to be (u,v)-revlex order. Then by (2.23),

$$\deg_{u,v} Q(x,y) \le \max\{\deg_{u,v}\phi_j(x,y) : j = 0, \cdots, C\} = \deg_{u,v}\phi_C(x,y) = r_A(C)$$

where $A = (a_K)$ is the sequence $\mathrm{Ind}(x^K)$, for (u,v)-revlex order. But $r_A(C) < \sqrt{2uvC}$ by a straightforward generalization of *Corollary 2*. ■

2.4.2 The Factorization Theorem

Lemma 1 *If* $f(x) \in F_v[x]$, *then* $\deg Q(x, f(x)) \leq \deg_{1,v} Q(x, y)$.

Proof: For $a_{i,j} \neq 0$, $\deg(x^i f(x)^j) \leq \deg Q(x^i x^{vj}) = i + vj \leq \max\{i + vj : a_{i,j} \neq 0\} = \deg_{1,v} Q(x, y)$. ■

Lemma 2 $Q(x, f(x)) = 0$ *if and only if* $(y - f(x))|Q(x, y)$.

Lemma 3 *If ord*$(Q : \alpha, \beta) = K$, *and* $f(\alpha) = \beta$, *then*

$$(x - \alpha)^K | Q(x, f(x)).$$

Proof: Using *Corollary 5* to express $Q(x, y)$ as a polynomial in $x - \alpha$ and $y - \beta$:

$$Q(x, y) = \sum_{i,j} b_{i,j}(x - \alpha)^i (y - \beta)^j$$

Then

$$Q(x, f(x)) = \sum_{i,j} b_{i,j}(x - \alpha)^i (f(x) - \beta)^j \tag{2.27}$$

since $f(\alpha) = \beta$, $f(x) - \beta$ is divisible by $x - \alpha$, so that the term $(x-\alpha)^i (f(x)-\beta)^j$ in (2.25) is divisible by $(x-\alpha)^{i+j}$. But ord $(Q : \alpha, \beta) = K$ implies that if $b_{i,j} \neq 0$, then $i + j \geq K$. Thus, every nonzero term in (2.25) is divisible by $(x - \alpha)^K$, i.e., $(x - \alpha)^K | Q(x, f(x)$. ■

Let $F_v[x]$ be polynomials of degree $\leq v$ from $F[x]$. If $Q(x, y) \in F[x, y]$, and $f(x) \in F[x]$. Define the Q-score of f as

$$S_Q(f) = \sum_{\alpha \in F} \text{ord}(Q : \alpha, f(\alpha))$$

Theorem 6 *Suppose* $f(x) \in F_v[x]$, $Q(x, y) \in F[x, y]$, *and*

$$S_Q(f) > deg_{1,v} Q$$

then $y - f(x)$ *is a factor of* $Q(x, y)$[4]

[4]Be careful the condition $\beta = f(\alpha)$

Proof: By *Lemma 3*, we know that $\prod_{\alpha \in F}(x-\alpha)^{\mathrm{ord}(Q:\alpha, f(\alpha))} | Q(x, f(x))$. But by Lemma 1, the degree of $Q(x, f(x))$ is (at most) $\deg_{1,v} Q(x, y)$, and the degree of $\prod_{\alpha \in F}(x - \alpha)^{\mathrm{ord}(Q:\alpha, f(\alpha))}$ is $S_Q(f)$. Thus, if $S_Q(f)$ exceeds $\deg_{1,v} Q$, it follows that $Q(x, f(x)) = 0$, and so by Lemma 2, $y - f(x)$ divides $Q(x, y)$. ■

An alternative statement of this theorem is

Lemma 4 *If $f(x)$ is a polynomial of degree at most v such that $\beta_i = f(\alpha_i)$ for at least t values of $i \in [n]$ and $mt > deg_{1,v} Q$, then $y - f(x)$ divides Q.*

2.5 A Second Look at the GS Algorithm

Armed with the preliminary material, we give a formal description and proof of the correctness of the GS algorithm in this section.

2.5.1 Prerequisite Notations and Concepts

To begin with, we list some of the technical details needed for a full discussion of the GS algorithm.

- $K(f, \beta) = |\{i : f(\alpha_i) = \beta_i\}|$, $D(f, \beta) = |\{i : f(\alpha_i) \neq \beta_i\}|$.
- $C(n, m) = n\binom{m+1}{2}$.
- $(1, v)-$revlex monomial order.
- The indices $A(K, v)$ =Ind(x^K) and $B(L, v)$ =Ind(y^L) (with respect to $(1, v)$-revlex order), with the rank of apparition functions

$$\begin{aligned} r_A(C) &= \max\{K : A(K, v) \leq C\} \\ r_B(C) &= \max\{L : B(L, v) \leq C\} \end{aligned}$$

- The numbers K_m, t_m, and L_m:

$$K_m(n,k) \triangleq 1 + \lfloor r_A(C)/m \rfloor \tag{2.28}$$

$$t_m(n,k) \triangleq n - K_m(n,k) = n - 1 - \lfloor r_A(C)/m \rfloor \tag{2.29}$$

$$L_m(n,k) \triangleq \max\{L : B(L,v) \le C(n,m)\} = r_B(C) \tag{2.30}$$

Proof: By (2.11), $\deg_{1,v} Q(x,y) \le \max\{\deg_{1,v}\phi_i(x,y) : i = 0, \cdots, C\} = r_A(C)$. By **Factorization Theorem**, any polynomial $f(x)$ of degree $\le v$ such that $mK(f,\beta) > r_A(C)$, will be a $y-$root of $Q(x,y)$. $\Rightarrow K(f,\beta) \ge 1 + \lfloor r_A(C)/m \rfloor = K_m$

- Estimates of K_m and L_m:

$$\left\lfloor \sqrt{vn\frac{m+1}{m}} - \frac{v}{2m} \right\rfloor + 1 \le K_m \le \left\lfloor \sqrt{vn\frac{m+1}{m}} \right\rfloor \tag{2.31}$$

$$n - \left\lfloor \sqrt{vn\frac{m+1}{m}} \right\rfloor \le t_m \le n - 1 - \left\lfloor \sqrt{vn\frac{m+1}{m}} - \frac{v}{2m} \right\rfloor \tag{2.32}$$

$$L_m = \left\lfloor \sqrt{\frac{n}{v}m(m+1) + \left(\frac{v+2}{2v}\right)^2} - \frac{v+2}{2v} \right\rfloor < \left(m + \frac{1}{2}\right)\sqrt{\frac{n}{v}} \tag{2.33}$$

2.5.2 Detailed GS Decoding Algorithm

Given an (n,k) RS code over the finite field F, with support set $(\alpha_1, \cdots, \alpha_n)$, and a positive integer m. The **input** of the GS(m) decoder is a vector of the form

$$\beta = (\beta_1, \cdots \beta_n) \in F^n$$

The **output** of the GS(m) decoder is a list of polynomials

$$\{f_1, \cdots, f_L\}$$

The GS(m) Decoder.

1. The GS(m) decoder constructs a nonzero two-variable polynomial of the form

$$Q(x,y) = \sum_{j=0}^{C(n,m)} a_j \phi_j(x,y)$$

where $\phi_0 < \phi_1 < \cdots$ is $(1, v)$-revlex monomial order, such that $Q(x, y)$ has a zero of order m at each of the n points (α_i, β_i), for $i = 1, \cdots, n$ [5].

2. The output of the algorithm is the list of y-roots of $Q(x, y)$, i.e.,

$$\mathcal{L} = \{f(x) \in F[x] : (y - f(x))|Q(x, y)\}$$

2.6 KÖtter's Solution to the Interpolation Problem

In general terms, the interpolation problem is to construct a bivariate polynomial $Q(x, y)$ with minimal $(1, v)-$degree that satisfies a number of constraints of the form

$$D_{r,s}Q(\alpha, \beta) = 0$$

where $(r, s) \in \mathbb{N}^2$ and $(\alpha, \beta) \in F^2$. It turns out that the mapping

$$Q(x, y) \mapsto D_{r,s}Q(\alpha, \beta)$$

is an example of what is called a *linear functional* on $F[x, y]$. It is no harder mathematically, and much easier notationally, to consider the more general problem of constructing a bivariate polynomial $Q(x, y)$ of minimal weighted-degree that satisfies a number of constraints of the form

$$D_iQ(x, y) = 0, \quad \text{for} \quad i = 1, 2, \cdots$$

where each D_i is a linear functional. The goal of this section is to describe an algorithm for solving the more general problem.

2.6.1 Linear Functionals on $F[x, y]$

Definition 3 *A mapping $D : F[x, y] \longrightarrow F$ is called a* **linear functional** *if*

$$D(\alpha P + \beta Q) = \alpha D(P) + \beta D(Q)$$

[5]The **Interpolation Theorem** guarantees that such a polynomial exists.

$\forall\ P, Q \in F[x,y], \alpha, \beta \in F$. *The kernel of a linear functional* D *is the set*

$$K = kerD = \{Q : D(Q) = 0\}$$

If D *is a linear functional with kernel* K*, the corresponding bi-linear mapping* $[P,Q]_D$ *is defined as*

$$[P,Q]_D \triangleq D(Q)P - D(P)Q \tag{2.34}$$

This simple mapping is a crucial part of the algorithms we present below; its key property is given in the following lemma.

Lemma 5 *For all* P, Q *in* $F[x,y]$, $[P,Q]_D \in kerD$.

Proof:‘ Let $\alpha = D(Q)$ and $\beta = D(P)$. Then $D([P,Q]_D) = D(\alpha P - \beta Q) = \alpha D(P) - \beta D(Q) = \alpha\beta - \beta\alpha = 0$, which proves $[P,Q]_D \in \ker D$. ■

2.6.2 Problem Statement

Let $F_L[x,y]$ denote the set of polynomials from $F[x,y]$ whose y-degree is $\leqslant L$, i.e., those of the form

$$Q(x,y) = \sum_{k=0}^{L} q_k(x) y^k$$

where each $q_k(x) \in F[x]$.

Let $D_1, \cdots, D_C$ be C linear functions defined on $F_L[x,y]$, and let $K_1, \cdots, K_C$ be the corresponding kernels, i.e.,

$$K_i = \{Q(x,y) \in F_L[x,y] : D_i(Q) = 0\}$$

The **cumulative kernels** $\overline{K}_0, \cdots, \overline{K}_C$ are defined as follows: $\overline{K}_0 = F_L[x,y]$ and for $i = 1, \cdots, C$,

$$\begin{aligned} \overline{K}_i &= \overline{K}_{i-1} \cap K_i \\ &= K_1 \cap \cdots \cap K_i \\ &= \{Q(x,y) \in F_L[x,y] : D_1(Q) = \cdots = D_i(Q) = 0\} \end{aligned}$$

The Generalized Interpolation Problem: Construct a **minimal** element from

$$\overline{K}_C = K_1 \cap \cdots \cap K_C$$

i.e., calculate

$$Q_0(x,y) \in \min\{Q(x,y) : D_1(Q) = \cdots = D_C(Q) = 0\}$$

where "minimal" means **minimal rank** (weighted degree) with respect to the given monomial order.

2.6.3 Kötter's Algorithm

The set of monomials from $F_L[x,y]$(the polynomials from $F[x,y]$ whose y-degree is $\leq L$), viz.,

$$\mathbb{M}_L[x,y] = \{x^i y^j : 0 \leq i, 0 \leq j \leq L\}$$

is partitioned according to the exponent of y: $\mathbb{M}_L[x,y] = \bigcup_{j=0}^{L} \mathbb{M}_j$, where

$$\mathbb{M}_j = \{x^i y^j : i \geq 0\}$$

The partition of $\mathbb{M}_L$ includes a partition on $F_L[x,y]$: $F_L[x,y] = S_0 \cup \cdots \cup S_L$, where

$$S_j = \{Q \in F_L[x,y] : LM(Q) \in \mathbb{M}_j\}$$

K0̈tter's algorithm generated a sequence of lists $G_0, G_1, \cdots, G_C$, with

$$G_i = (g_{i,0}, \cdots, g_{i,L})$$

where $g_{i,j}$ is a minimal element of $\overline{K}_i \cap S_j$. The algorithm's output is the polynomial

$$Q_0(x,y) = \min_{0 \leq j \leq L} g_{C,j}(x,y)$$

which is a minimal element of $\overline{K}_C$.

K0̈tter's algorithm is initialized as follows:

$$g_{0,j} = y^j, \ j = 0, \cdots, L$$

Given G_i, i.e., $\{g_{i,j}\}_{j=0}^{L}$, G_{i+1} is defined recursively:

$$\begin{aligned} J_0 &= \{j : D_{i+1}(g_{i,j}) = 0\} \\ J_1 &= \{j : D_{i+1}(g_{i,j}) \neq 0\} \end{aligned}$$

If J_1 is not empty,among the polynomials $g_{i,j}$ with $j \in J_1$, let g_{i,j^*} be the one with minimal rank, and temporarily denote g_{i,j^*} by f:

$$\begin{aligned} f &= \min_{j \in J_1} g_{i,j} \\ j^* &= \arg\min_{j \in J_1} g_{i,j} \end{aligned}$$

Then using the notation of (2.32), $g_{i+1,j}$ is defined for $j = 0, \cdots, L$:

$$g_{i+1,j} = \begin{cases} g_{i,j} & \text{if } j \in J_0 \\ [g_{i,j}, f]_{D_{i+1}} & \text{if } j \in J_1 \text{ but } j \neq j^* \\ [xf, f]_{D_{i+1}} & \text{if } j = j^* \end{cases}$$

Theorem 7 [3] *For $i = 0, \cdots, C$, we have*

$$g_{i,j} = \min\{g : g \in \overline{K}_i \cap S_j\} \quad \textit{for } j = 0, \cdots, L$$

Moreover, for GS decoding,

- Given L, $(\alpha_i, \beta_i)_{i=1}^{n}$, $(m_i)_{i=1}^{n} = m$, $(1, k-1)$ weighted degree monomial order.
- $D_{r,s} g_j(\alpha_i, \beta_i) = 0$, for $0 \leq r + s < m$, $1 \leq i \leq n$, $0 \leq j \leq L$.

2.7 The Roth-Ruckenstein Solution to the Factorization Problem

In this section, we present the most efficient algorithm currently known for solving the factorization problem, due to Roth and Ruckenstein (RR) [4].

The factorization problem can be stated as follows. Given a polynomial $Q(x, y) \in F[x, y]$, find all polynomials $f(x)$ of degree $\leq v$ such that $(y - f(x)) | Q(x, y)$. Alternatively, find all $f(x) \in F_v[x]$ such that

$$Q(x, f(x)) \equiv 0 \tag{2.35}$$

If (2.33) holds, we call $f(x)$ a y-root of $Q(x,y)$. In this section, we will describe the RR algorithm for finding $y-$roots.

For a two-variable polynomial $Q(x,y)$ such that $x^m|Q(x,y)$, but $x^{m+1} \nmid Q(x,y)$, we define

$$\langle Q(x,y)\rangle = \frac{Q(x,y)}{x^m}.$$

Although $Q(0,y)$ might be identically zero, nevertheless $\langle Q(0,y)\rangle$ is a nonzero polynomial in y (e.g., if $Q(x,y) = xy$, $Q(x,y) = 0$ but $\langle Q(0,y)\rangle = y$).

Suppose

$$f(x) = a_0 + a_1 x + \cdots + a_v x^v$$

is a $y-$root of $Q(x,y)$. We will see that the coefficients $a_0, a_1, \cdots, a_v$ can be *picked off* one at a time. As a start, the following lemma shows how to determine a_0.

Lemma 6 *If $(y - f(x))|Q(x,y)$, then $y = f(0) = a_0$ is a root of the equation*

$$Q_0(0,y) = 0$$

where $Q_0(x,y) = \langle Q(x,y)\rangle$ and $f(x) = a_0 + a_1 x + \cdots + a_v x^v$.

Proof: By definition, $Q(x,y) = x^m Q_0(x,y)$ for some $m \geq 0$. Thus, if $(y-f(x))|Q(x,y)$, then $(y-f(x))|Q_0(x,y)$ as well, so that $Q_0(x,y) = (y-f(x))T_0(x,y)$ for some polynomial $T_0(x,y)$. Thus, $y = f(0)$ is a solution of the equation $Q_0(0,y) = 0$.

We now proceed by induction, defining three sequences of polynomials, $f_j(x)$, $T_j(x,y)$, and $Q_j(x,y)$, for $j = 0, 1, \cdots, v$, as follows.

Initialization: $f_0 := f(x)$, $Q_0(x,y) := \langle Q(x,y)\rangle$.

For $j \geq 1$, define

$$f_j(x) := (f_{j-1}(x) - f_{j-1}(0))/x = a_j + \cdots + a_v x^{v-j} \tag{2.36}$$

$$T_j(x,y) := Q_{j-1}(x, xy + a_{j-1}) \tag{2.37}$$

$$Q_j(x,y) := \langle T_j(x,y)\rangle \tag{2.38}$$

■

Theorem 8 *Given* $f(x) = a_0 + a_1x + \cdots + a_vx^v \in F_v[x]$, *and* $Q(x,y) \in F[x,y]$. *For any* $j \geq 1$, $(y - f(x))|Q(x,y)$ *if and only if* $(y - f_j(x))|Q_j(x,y)$. *If* $j \geq 1, (y - f_j(x))|Q_j(x,y) \leftrightarrow (y - f_{j-1}(x))|Q_{j-1}(x,y)$

For detailed proof see [3].

Here is the "*picking off*" theorem.

Corollary 9 *If* $(y - f(x))|Q(x,y)$, *then* $y = a_j$ *is a root of the equation*

$$Q_j(0,y) = 0, \quad \textit{for } j = 0, \cdots, v$$

Proof: By **Theorem 8**, $y - f_j(x)$ divides $Q_j(x,y)$ for all $j \geq 0$. Substituting $x = 0$ yields the stated result, since $f_j(0) = a_j$. ■

Stopping Criterion:

Corollary 10 *If* $y|Q_{v+1}(x,y)$, *i.e., if* $Q_{v+1}(x,0) = 0$, *then* $f(x) = a_0 + \cdots + a_vx^v$ *is a y-root of* $Q(x,y)$.

Proof: (2.34) implies that $f_j(x) = 0$ for all $j \geq v+1$, so that the hypothesis $y|Q_{v+1}(x,y)$ says that $(y - f_{v+1}(x))|Q_{v+1}(x,y)$. Invoking *Theorem 8*, we have $j \geq 1, (y - f_j(x))|Q_j(x,y) \leftrightarrow (y - f_{j-1}(x))|Q_{j-1}(x,y) \Rightarrow (y - f(x))|Q_0(x,y)$. ■

Chapter 3

Algebraic Soft-Decision Decoding of the GS Algorithm

Fig. 2.1 has shown that for high rate RS codes, the error-correcting capability can not be improved anymore. Therefore, we need to do soft-decision decoding to improve this situation. In the following section we show how the soft-decision reliability information provided by the channel should be translated into algebraic interpolation conditions. A block diagram of soft-decision algorithm is given in Fig. 3.1.

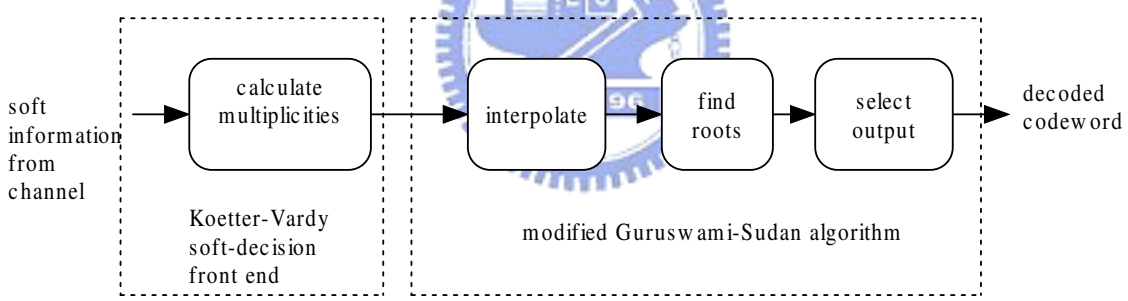


Figure 3.1: The KÖtter-Vardy algorithm

3.1 Algebraic Soft-Decision Decoding

For RS codes, let $\alpha_1, \alpha_2, \cdots, \alpha_q$ be q distinct element in $GF(q)$. We think of channel input and output as random variables X and Y, respectively. Given the received vector $\underline{y} = (y_1, y_2, \cdots, y_n)$ observed at the channel output, we compute, for $i = 1, 2, \cdots, q$ and

$j = 1, 2, \cdots, n$,

$$\pi_{i,j} \triangleq Pr(X = \alpha_i | Y = y_j). \tag{3.1}$$

Let Π be the $q \times n$ matrix with entries $\pi_{i,j}$ defined in (3.1). 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 jth column of Π in the row indexed by $\alpha \in \mathrm{GF}(q)$.

The first step in hard-decision decoding is the construction of the hard-decision vector $\underline{u} = (u_1, u_2, \cdots, u_n) \in F_q^n$, where

$$u_j \triangleq \arg \max_{\alpha \in GF(q)} \Pi(\alpha, j), \quad \text{for} \quad j = 1, 2, ..., n$$

This hard-decision vector is then taken as the channel output $\underline{c} + \underline{e}$, 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 somehow convert these probabilities into algebraic conditions. The algebraic soft-decision RS decoder developed in this section converts the reliability matrix Π into a choice of interpolation points and their multiplicities in the Guruswami-Sudan list-decoding algorithm.

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 $m_{i,j}$. Thus the first step of our decoding algorithm consists of computing the multiplicity matrix M from the reliability matrix Π. This step is discussed in detail in the next section. The second step consists of the following.

Soft interpolation step: Given the point set and the multiplicity matrix $M = [m_{i,j}]$, compute a nontrivial bivariate polynomial $Q_M(X, Y)$ of minimal (1,v)-revlex 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 step of the algorithm is the factorization step, which is identical to the factorization step(will be described later) of the Guruswami-Sudan algorithm.

In the following, we characterize the conditions under which the decoder will produce the transmitted codeword, for a given choice of interpolation points and their multiplicities (that is, for a given multiplicity matrix M).

Definition 4 *Given a $q \times n$ matrix M with nonnegative integer entries $m_{i,j}$, we define the cost of M as follows:*

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

It is easy to see that the computation of $Q_M(X,Y)$ is equivalent to solving a system of linear equations of (2.21), like the GS algorithm interpolation. Since a given zero of multiplicity m imposes $m(m+1)/2$ linear constraints on the coefficients of $Q_M(X,Y)$, the cost $C(M)$ is precisely the total number of linear equations. We can always find a nonzero solution $Q_M(X,Y)$ to the soft interpolation task if the $\deg Q_{1,v}$ is large enough. That is

$$N_{w_x,w_y}(\delta) \triangleq |\{X^i Y^j : i,j \geqslant 0 \text{ and } iw_x + jw_y \leqslant \delta|$$

if

$$N_{1,k-1}(\delta) > C(M) \tag{3.2}$$

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

$$\triangle_{w_X,w_Y}(\nu) \triangleq \min\{\delta \in \mathbb{Z} : N_{w_X,w_Y}(\delta) > \nu\}. \tag{3.3}$$

Next, given two $q \times n$ matrices A and B over the same field, we define the inner product

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

Finally, it will be convenient to think of the codewords of the RS code $\mathbb{C}_q(n,k)$ as $q \times n$ matrices over the reals. Specifically, any vector $\underline{v} = (v_1, v_2, \cdots, v_n)$ over $GF(q)$ can be represented by the $q \times n$ real-valued matrix 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 5 *The score of a vector* $\underline{v} = (v_1, v_2, \cdots, v_n)$ *over* $GF(q)$ *with respect to a given multiplicity matrix* M *is defined as the inner product* $S_{\mathrm{M}}(\underline{v}) = \langle M, [\underline{v}] \rangle$.

The following theorem characterizes the set of codewords produced by our soft-decision decoding algorithm for a given multiplicity matrix.

Theorem 9 *Let* C *be the cost of a given multiplicity matrix* M . *Then the polynomial* $Q_{\mathrm{M}}(X,Y)$ *has a factor* $Y - f(X)$, *where* $f(X)$ *evaluates to a codeword* $\underline{c}$ *with degree* v , *if the score of* $\underline{c}$ *is large enough, namely, if*

$$S_M(\underline{c}) > \triangle_{1,k-1}(C)$$

The proof is similar to the GS algorithm factorization theorem.(see theorem6.)

3.2 From Posterior Probabilities to Interpolation Points

This section develops an algorithm that converts 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 (3.1), we compute the multiplicity matrix M that serves as input to the soft interpolation step. Let $\mathcal{M}_{q,n}$ denote the set of all $q \times n$ matrices with nonnegative integer entries $m_{i,j}$, and let $\mathcal{M}(C)$ be the finite set of all matrices in $\mathcal{M}_{q,n}$ whose cost is equal to C. Thus

$$\mathcal{M}(C) \triangleq \{M \in \mathcal{M}_{q,n} : \frac{1}{2}\sum_{i=1}^{q}\sum_{j=1}^{n} m_{i,j}(m_{i,j}+1) = C\}$$

In view of **Theorem 9**, we would like to choose $M \in \mathcal{M}(C)$ so as to maximize the score of the transmitted codeword $\underline{c} \in \mathbb{C}_q(n,k)$.

As far as the decoder is concerned, the transmitted codeword may be thought of as a random vector $\chi = (\chi_1, \chi_2, \cdots, \chi_n)$. Thus $S_M(\chi)$ is a random variable, and the question is: what is the best choice of a multiplicity matrix $M \in \mathcal{M}(C)$ in this probabilistic setting? We choose to compute the matrix $M \in \mathcal{M}(C)$ that directly maximizes the expected value of $S_M(\chi)$.

To proceed, let us define the **expected score** with respect to a probability distribution $P(\cdot)$ on the random vector $\chi = (\chi_1, \chi_2, \cdots, \chi_n)$ as follows:

$$E_P\{S_M(\chi)\} \triangleq \sum_{\underline{x} \in F_q^n} S_M(\underline{x}) P(\underline{x}) = \sum_{\underline{x} \in F_q^n} \sum_{j=1}^{n} M(x_j, j) P(\underline{x}), \tag{3.4}$$

where $M(x_j, j)$ denotes the entry found in the jth 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 $(y_1, y_2, \cdots, y_n)$, namely

$$P(x_1, x_2, \cdots, x_n) \triangleq \prod_{j=1}^{n} \Pr(X_j = x_j | Y_j = y_j) = \prod_{j=1}^{n} \Pi(x_j, j) \tag{3.5}$$

where Π is the reliability matrix defined in Eq.(3.1). It is easy to see that this would be the a posteriori distribution of χ given the channel observations.

Thus, we want to find $M(\Pi, C)$ defined as follows:

$$M(\Pi, C) \triangleq \text{argmax}_{M \in \mathcal{M}(C)} E_P\{S_M(\chi)\}$$

where the expectation is taken with respect to the probability distribution $P(\cdot)$ in Eq.(3.5). We start with the following lemma, which gives a useful expression for the expected score.

Lemma 7 *The expected score with respect to the probability distribution of (3.5) is equal to the inner product of the multiplicity matrix and the reliability matrix, namely*

$$E_P\{S_M(\chi)\} = \langle M, \Pi \rangle$$

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 $\pi^*_{i,j} := \frac{\pi_{i,j}}{m_{i,j}+2}$, $m_{i,j} := m_{i,j}+1$, $s := s-1$
Control step: If $s = 0$, return M; otherwise go to the iteration step.

Table 3.1: Algorithm A

Proof: It is easy to see that if χ is distributed according to (3.5), then Π is precisely the componentwise expected value of $[\chi]$. The lemma now follows by linearity of expectation

$$E_P\{S_M(\chi)\} = E_P\{\langle M, [\chi]\rangle\} = \langle M, E_P\{[\chi]\}\rangle = \langle M, \Pi\rangle$$

■

We will construct $M(\Pi, C)$ iteratively, starting with the all-zero matrix and increasing one of the entries in the matrix at each iteration. Referring to Lemma 5, 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 $Q_M(X,Y)$ 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. These observations lead to **Algorithm A**, which greedily maximizes the ratio of the increase in the expected score to the increase in cost at each iteration.

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. The following theorem shows that this matrix is optimal.

Theorem 10 *The matrix $\mathcal{M}(\Pi, s)$ maximizes the expected score among all matrices in*

$\mathbb{M}_{q,n}$ *with the same cost. That is, if* C *is the cost of* $\mathcal{M}(\Pi, s)$*, then*

$$\mathcal{M}(\Pi, s) = argmax_{M \in \mathbb{M}(C)} \langle M, \Pi \rangle$$

Proof: With each position (i, j) in the reliability matrix Π, we associate an infinite sequence of rectangles $\mathcal{B}_{i,j,1}, \mathcal{B}_{i,j,2}, \cdots$ indexed by the positive integers. Let $\mathcal{B}$ denote the set of all such rectangles. For each rectangle $\mathcal{B}_{i,j,l} \in \mathcal{B}$, we define its length$(\mathcal{B}_{i,j,l}) = l$, height$(\mathcal{B}_{i,j,l}) = \pi_{i,j}/l$, and

$$area(\mathcal{B}_{i,j,l}) = length(\mathcal{B}_{i,j,l}) \cdot height(\mathcal{B}_{i,j,l}) = \pi_{i,j}$$

For a multiplicity matrix $M \in \mathbb{M}_{q,n}$, we define the corresponding set of rectangles

$$\vartheta(M) \triangleq \{\mathcal{B}_{i,j,l} : 1 \leqslant i \leqslant q, 1 \leqslant j \leqslant n, 1 \leqslant l \leqslant m_{i,j}\}$$

Note that the number of rectangles in $\vartheta(M)$ 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. Furthermore

$$\begin{aligned}
C(M) &= \sum_{i=1,j=1}^{q,n} \frac{m_{i,j}(m_{i,j}+1)}{2} = \sum_{i=1,j=1}^{q,n} \sum_{l=1}^{m_{i,j}} l \\
&= \sum_{i=1,j=1}^{q,n} \sum_{l=1}^{m_{i,j}} length(\mathcal{B}_{i,j,l}) = \sum_{\mathcal{B} \in \vartheta(M)} length(\mathcal{B}) \\
\langle M, \Pi \rangle &= \sum_{i=1,j=1}^{q,n} m_{i,j} \cdot \pi_{i,j} = \sum_{i=1,j=1}^{q,n} \sum_{l=1}^{m_{i,j}} \pi_{i,j} \\
&= \sum_{i=1,j=1}^{q,n} \sum_{l=1}^{m_{i,j}} area(\mathcal{B}_{i,j,l}) = \sum_{\mathcal{B} \in \vartheta(M)} area(\mathcal{B}).
\end{aligned}$$

Thus the cost of M is the total length of all the rectangles in $\vartheta(M)$ and the expected score $\langle M, \Pi \rangle$ is the total area of all the rectangles in $\vartheta(M)$. 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 $\mathcal{B}$. Indeed, it is easy to see that the ratios $\pi^*_{i,j}$ with which **Algorithm A** operates are precisely the heights of the rectangles.

The algorithm *removes* from $\mathcal{B}$ and puts in $\vartheta(M)$ the highest rectangle available at each iteration. It is now obvious that if the s highest rectangles in $\mathcal{B}$ have total length C, then no collection of rectangles of total length at most C can have a larger total area. ■

Chapter 4

Reduced-Complexity Interpolation-Based Soft-Decision RS Decoders

The KV soft-decision decoding procedure results in very impressive error correcting performance. Nevertheless, the algorithm is still quite computationally complex and not straightforward to implement. This chapter introduces algorithmic techniques that reduce the complexity of interpolation-based decoders to the point where efficient software or VLSI implementations are possible.

4.1 Modified Interpolation

The interpolation algorithm is the most time-consuming component of KV decoding and it is essential to reduce its complexity if KV decoding is to be used in real-time applications. From the discussion in section (2.4), if a multiplicity matrix has a maximum entry m, then the maximum interpolation cost would be the cost of hard-decision GS decoding with multiplicity m:

$$C = (\frac{n}{2})m(m+1)$$

. For the remainder of this discussion, we will assume the worst-case where the cost of interpolation is the maximum possible cost, C. The interpolation algorithm needs to store $(d_y + 1)$ bivariate polynomials.(d_y is the maximal $y-$degree) Since a homogeneous

linear system must have more unknowns than equations, the length (number of terms) of the polynomials must be at least C. The memory requirements of interpolation are $\approx (d_y + 1)C$ field elements. Therefore the complexity of interpolation in terms of the number of Galois field operations is $N_{\text{op,interp}} = O(d_y C^2)$.

Example 10 *Consider a decoder for a (255,239) RS code with maximum multiplicity* $m = \{1, 4, 16\}$. *The interpolation cost, complexity and memory requirements are summarized in Table (4.1).*

m	C	$\text{N}_{op,interp}$	memory
1	255	1×10^5	512 bytes
4	2550	3×10^7	12 Kbytes
16	34680	2×10^{10}	576 Kbytes

Table 4.1: The maximum cost, complexity ($\text{N}_{\text{op,interp}}$) and memory requirements for the interpolation algorithm applied to a RS(255,239) code with maximum multiplicity m. The data is from [5]

¿From Table(4.1) we see that the decoding complexity and memory requirements grow very quickly as the multiplicity increases. If the maximum multiplicity is fixed to deliver a desired error-rate, then to lower the cost and hence the complexity, the number of interpolation points (nonzero entries in the multiplicity matrix) that we apply the bivariate interpolation algorithm to must be reduced. We apply the trick of "reencoding" the received word to reduce the interpolation complexity.

4.1.1 Systematic Encoding

A systematic encoding is one where all the k input symbols to the encoder explicitly appear in the encoded codeword. If they appear in k consecutive positions then the encoding is called strictly systematic. Strictly systematic encoders, where the k message symbols appear as the last k symbols in a codeword, are easily implemented with a linear feedback shift register [6] and are commonly used.

Interpolation-based decoding algorithms rely on an evaluation map encoding, however it is more efficient to implement an encoder as a linear feedback shift register. We would also like to apply the decoder to existing RS transmission systems that use a systematic encoder. Therefore we would like to use a systematic encoding in place of the evaluation map encoding.

We are also interested in generating systematic encodings where the information appears in arbitrary positions in an encoded codeword. If these positions can change for every encoder use, then an efficient way of implementing this systematic encoder is with an erasures-only RS decoder [6]. Since RS codes are minimum-distance-separable (MDS), a codeword may be perfectly recovered from any k of its symbols. An erasures-only decoder is much simpler than an error-and-erasures decoder since the erasure locations are known a-priori. Therefore, the expensive iterative Berlekamp-Massey algorithm for solving the key equation and the Chien search root-finding can be skipped .

4.1.2 Re-encoding

The idea of re-encoding is to transform the interpolation problem into one that is easier to solve. The codeword c is transmitted through a noisy channel. The hard decision vector, $r = (r_0, r_1, \cdots, r_{n-1})$, which can be extracted from the reliability matrix Π, is $r = c + e$, where e is an error vector. The first step is to partition the received symbols in r into two sets, U ("unreliable") and R ("reliable"). The set R consists of the k most reliable symbols, where the reliability information can be derived from the reliability matrix Π. The set of positions of the symbols in R(labeled from 1 to n corresponding to $\{\alpha^1, \alpha^2, \cdots, \alpha^n\}$) is the set R_k. Now systematically encode the symbols in R so that they appear in the reencoded codeword, ψ,in the same positions that they appeared in r. As discussed in Section (4.1.1), this can be done efficiently for k arbitrary

positions with an erasures-only decoder. Taking the difference between r and ψ we get:

$$\begin{aligned} r' &= r - \psi && (4.1)\\ &= (c + e) - \psi && (4.2)\\ &= (c - \psi) + e && (4.3) \end{aligned}$$

which is a codeword (by the linearity of the code) that is corrupted by the same error pattern as r. However, r' has a very interesting property; since the reencoding of r is systematic, k symbols of r' are zero. These zero symbols correspond to k interpolation points with a zero $y-$component:

$$V = \{(\alpha^i, 0)\}, \qquad i \in R_k \tag{4.4}$$

An interpolation polynomial for the k points in V is $v(x)^m$ where $v(x)$ is found through a simple univariate interpolation:

$$v(x) = \prod_{i \in R_k} (x - \alpha^i) \tag{4.5}$$

The advantage for high-rate codes is that we have found an interpolation polynomial for most of the points without having to use the expensive bivariate interpolation algorithm. The calculation of $v(x)$ requires a single polynomial to be updated k times instead of $(d_y + 1)$ polynomials being updated $(k/2)(m^2 + m)$ times. Now that we have $v(x)$, the bivariate interpolation algorithm is run at decode time starting from the initial polynomial set:

$$G = \{v(x)^m, v(x)^{m-1}y, ..., v(x)^{m-d_y}y^{d_y}\} \tag{4.6}$$

and run for at most

$$\begin{aligned} C' &= (\frac{n-k}{2})(m)(m+1)\\ &= (1 - \frac{k}{n})C \end{aligned}$$

iterations, where C is the maximum cost of the bivariate interpolation without reencoding. The reduced cost, C', gets smaller as the code rate k/n increases.

4.1.3 Reducing the Memory Requirements

The memory requirements for interpolation can be very large since the maximum length of the bivariate polynomials is at least C terms. The polynomials can be shortened by factoring out the polynomial $v(x)$. It is shown in [7],[8] that if reencoding is used then the interpolation polynomial $P(x, y)$ can be written as:

$$P(x,y) = \sum_{j=0}^{d_y} w_j(x) \prod_{i \in R_k} (X - \alpha^i)^{m_i - j} T_j(x) y^j \tag{4.7}$$

where,

$$T_j(x) = \prod_{i \in R_k} (x - \alpha^i)^{\max(j - m_i, 0)} \tag{4.8}$$

This is the most general way to decompose $P(x, y)$ and it allows any k symbols to be chosen for R. However, it seems that the most logical choice for R is to choose the k symbols with the largest reliability (largest multiplicities) to achieve the maximum complexity reduction. To obtain the shortest possible polynomials, we make the assumption that R consists of k points that have the maximum possible multiplicity $m = d_y$. Then $T_j(x) = 1$, $j = 0, \cdots, d_y$ and Eq.(4.7) reduces to [10]:

$$P(x,y) = \sum_{j=0}^{d_y} w_j(x) v(x)^{m-j} y^j \tag{4.9}$$

which means that common factors of $v(x)$ are being carried around needlessly, wasting memory. It would be nice to factor out the powers of $v(x)$ and only have to calculate the $w_j(x)$ in real-time. The interpolation polynomial can be written as

$$P(x,y) = \sum_{j=0}^{d_y} w_j(x) v(x)^{m-j} y^j \tag{4.10}$$

$$= v^m(x) \sum_{j=0}^{d_y} w_j(x) \left(\frac{y}{v(x)}\right)^j \tag{4.11}$$

The decoding algorithm takes the transformed word r' as input and tries to estimate the transformed codeword $c' = c - \psi$. Therefore, if the decoding is successful, a message

polynomial $f'(x)$ corresponding to c' will be a linear $y-$root of $P(x,y)$, i.e.:

$$P(x,y) = (y - f'(x))A(x,y) \tag{4.12}$$

or,

$$P(x, f'(x)) = 0 \tag{4.13}$$

From Eq.(4.11),

$$v^m(x)\sum_{j=0}^{d_y} w_j(x)(\frac{f'(x)}{v(x)})^j = 0 \tag{4.14}$$

$$\sum_{j=0}^{d_y} w_j(x)(\frac{f'(x)}{v(x)})^j = 0 \tag{4.15}$$

Define the reduced interpolation polynomial :

$$\widetilde{P}(x,\widetilde{y}) = \sum_{j=0}^{d_y} w_j(x)\widetilde{y}^j \tag{4.16}$$

where $\widetilde{y} = y/v(x)$. Then if $f'(x)$ is a linear $y-$root of $P(x,y)$ it follows that $f'(x)/v(x)$ is a linear $\widetilde{y}$-root of the reduced interpolation polynomial $\widetilde{P}(x,\widetilde{y})$. A simplified interpolation can be carried out to find $\widetilde{P}(x,\widetilde{y})$ which is much shorter than $P(x,y)$ since the degree k polynomial $v(x)$ has been factored out in advance. To implement the simplified interpolation, consider the original set of polynomials, $G = \{1, y, \cdots, y^{d_y}\}$. After applying the reencoding technique, the starting polynomial set for decoding is:

$$\begin{aligned} G' &= \{v(x)^m, v(x)^{m-1}y, \cdots, v(x)^{m-d_y}y^{d_y}\} && (4.17)\\ &= v(x)^m\{1, \frac{y}{v(x)}, (\frac{y}{v(x)})^2, \cdots, (\frac{y}{v(x)})^{d_y}\} && (4.18)\end{aligned}$$

After a change of variables $\widetilde{y} = y/v(x)$, we have $\widetilde{G} = \{1, \widetilde{y}, \cdots, \widetilde{y}^{d_y}\}$. Note that the weighted degree of the new variable $\widetilde{y}$ is :

$$\begin{aligned} \deg^{(1,k-1)}(\widetilde{y}) &= \deg^{(1,k-1)}(y) - \deg^{(1,k-1)}(v(x)) \\ &= (k-1) - k \\ &= -1. \end{aligned}$$

The y-coordinates of the interpolation points need to be rescaled:

$$\widetilde{y}_i = \frac{y'_i}{v(x^i)} \tag{4.19}$$

where the y'_i are the y-coordinates of points after the translation in the reencoding step. Starting from $\widetilde{G} = \{1, \widetilde{y}, \cdots, \widetilde{y}^{d_y}\}$, one applies the KV interpolation algorithm to the $O(n-k)$ rescaled points where the min function is taken with respect to the (1,-1)-weighted degree of the polynomials in x and $\widetilde{y}$. A formal proof that simplified interpolation produces a correct result is given in [7] where the factorization step is slightly modified. This will be discussed in the following.

However, from (4.9), the simplified interpolation is not always realizable in all cases. It must satisfy

Lemma 8 *The maximum y-degree is equal to m.*

4.2 Reduced-Complexity Factorization

The savings realized by simplified interpolation can be carried over into the factorization procedure by applying the RR algorithm directly to the reduced polynomial $\widetilde{P}(x, \widetilde{y})$ as proposed in [7],[8],[14]. If the message polynomial corresponding to $c' = (c - \psi)$ is a linear y-root of $P(x, y)$ then $f'(x)/v(x)$ is a linear $\widetilde{y}$-root of $\widetilde{P}(x, \widetilde{y})$, or

$$\widetilde{P}(x, \widetilde{y}) = (\widetilde{y} - \frac{f'(x)}{v(x)})B(x, \widetilde{y}) \tag{4.20}$$

Applying the RR algorithm to the reduced polynomial, we obtain a sequence $s_0, s_1, \cdots, s_{l-1}$ (as a rule of thumb, we use $l = 2\lceil (k/n)t \rceil$) which are the coefficients of

$$s(x) = \frac{f'(x)}{v(x)}$$

The transformed received hard-decision word is $r' = c' + e$, which has zeroes in the k re-encoded positions, R_k. Therefore, in the k re-encoded positions, $c'_i = -e_i$, $i \in R_k$, or

,

$$f'(\alpha^i) = -e_i, \quad i \in R_k$$

If there is no error in position $i \in R_k$ then $e_i = 0$ and $f'(\alpha^i) = 0$. Therefore $(x - \alpha^i)$ is a root of $f'(x)$, or $f'(x) = (x - \alpha^i)D(x)$. Considering all the error-free positions in R_k,

$$f'(x) = \prod_{i \in R_k \quad s.t. \ e_i = 0} (x - \alpha^i)\Omega(x) \tag{4.21}$$

Therefore,

$$\begin{aligned} s(x) &= \frac{f'(x)}{v(x)} && (4.22) \\ &= \frac{\prod_{i \in R_k \quad s.t. \ e_i = 0}(x - \alpha^i)\Omega(x)}{\prod_{i \in R_k}(x - \alpha^i)} && (4.23) \\ &= \frac{\Omega(x)}{\prod_{i \in R_k \ s.t \ e_i \neq 0}(x - \alpha^i)} && (4.24) \end{aligned}$$

The denominator is an error-locator polynomial for the k positions in R_k. Given the *syndrome* polynomial $s(x)$, we can use the Berlekamp-Massey algorithm to reconstruct the rational function $\Omega(x)/\Lambda(x)$ where $\Lambda(x)$ is an error-locating polynomial for the k positions in R_k and $\Omega(x)$ is an error-evaluator polynomial for the k positions in R_k. The roots of $\Lambda(x)$ give the error locations in R_k . This technique only finds errors in the set of k reliable positions and not in the $(n - k)$ unreliable positions. To correct any errors in the $(n - k)$ unreliable positions, we can do a systematic reencoding in k arbitrary positions using the erasures-only decoder that is already implemented for the reencoding step.

We are only directly correcting errors in the k reliable positions. Fortunately, most errors are likely in the $(n - k)$ unreliable positions so we only need to correct a small number of errors and hence only need a few coefficients in the syndrome sequence. This greatly speeds up the Roth-Ruckenstein algorithm. As a rule of thumb we use $l = 2\lceil (k/n)t \rceil$ coefficients where $t = \lfloor (n-k)/2 \rfloor$ is the classical error-correcting capability. To compute the error values, we invoke (4.22) to have $e_i = -f'(\alpha^i), i \in R_k$ and

$$\begin{aligned} \tfrac{f'(x)}{v(x)} &= \frac{\Omega x}{\Lambda x} && (4.25) \\ f'(x) &= \frac{\Omega(x)v(x)}{\Lambda(x)} && (4.26) \end{aligned}$$

When evaluating $f'(x)$ at x-values corresponding to the error locations, $v(\alpha^i) = 0$ and $\lambda(\alpha^i) = 0$ where i is an error position in R_k. Using the L'Hopital's rule, we obtain the error-evaluation formula

$$f'(\alpha^i) = \frac{\Omega(\alpha^i)v^{(1)}(\alpha^i)}{\Lambda^{(1)}(\alpha^i)} \tag{4.27}$$

for the x-values α^i corresponding to error positions in R_k, where $v^{(1)}(x)$ and $\Lambda^{(1)}(x)$ are the formal derivatives of $v(x)$ and $\Lambda(x)$. Notice that we have directly found an estimate of the error vector, $\widehat{e}$, without having to subtract off r'. The estimated codeword can be found by adding $\widehat{e}$ to the received word r and the message can be read off directly if a systematic encoder was used. A block diagram of this scheme is shown in Fig. 4.1.

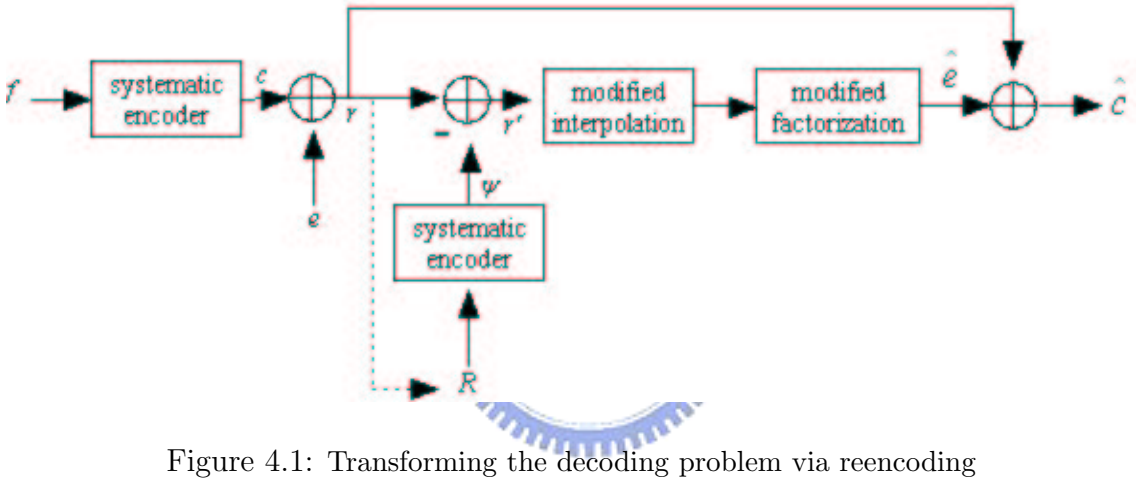


Figure 4.1: Transforming the decoding problem via reencoding

Chapter 5

Serially concatenated Convolutional and RS codes

In this chapter, we discuss the serially concatenated convolutional and RS codes with an emphasis on soft input and soft output algorithms. Although GS algorithm is an SD decoding algorithm it cannot provide soft output value. We therefore introduce the SSID algorithm [21] (stochastic shifting based iterative decoding) which does yield soft output values.

5.1 Stochastic Shifting Based Iterative Decoding (SSID) of RS codes

Consider a narrow sense (n, k) RS code over $GF(q^m)$, $n = q^m - 1$, which has a minimum distance $\delta = n - k + 1$. The parity check matrix can be represented by

$$H = \begin{pmatrix} 1 & \beta & \beta^2 & \cdots & \beta^{(n-1)} \\ 1 & \beta^2 & \beta^4 & \cdots & \beta^{2(n-1)} \\ & & & \cdots & \\ 1 & \beta^2 & \beta^4 & \cdots & \beta^{2(n-1)} \end{pmatrix}. \tag{5.1}$$

Here we consider RS codes over an extension field of GF(2). Let β be a primitive element in GF(2^m), all the 2^m elements in GF(2^m), $0, 1, \beta, \beta^2, \cdots, \beta^{2^m-2}$, can be represented using a binary vector expansion in $GF(2)$. Let $p(x)$ be a primitive polynomial in $F_2[x]$ and C be its companion matrix. The companion matrix is an $m \times m$ binary matrix. Since the mapping $\beta^i \longleftrightarrow C^i, \{i = 0, 1, 2, \cdots\}$ induces a field isomorphism, a binary

parity-check matrix $\mathcal{H}$ is obtained by replacing every element β^i in the parity check matrix H by its corresponding matrix C^i.

Example 11 *For GF(32), the primitive polynomial $p(x)$ is $x^5 + x^2 + 1$, then*

$$1 = \beta^0 \longleftrightarrow \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

$$\beta^1 \longleftrightarrow \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Given the channel output samples, the a posteriori log-likelihood ratio (LLR) and soft symbol values can be derived by using the belief propagation (BP) algorithm. However, standard BP does not work well for high density parity check codes (HDPC) codes due to *error propagation*.

By taking advantage of the cyclic property of RS codes, a sum product algorithm (SPA) with a stochastic shifting schedule is proposed [21, 22] to help alleviate the deterministic errors. Let $L^{(j)}$ denote the sum of the received LLRs and all extrinsic LLR produced until the jth iteration. During the jth iteration, the SPA is used on the vector $L^{(j)}$ to produce extrinsic information $L^{(j)}_{ext}$. The LLR $L^{(j+1)}$ is then updated according to:

$$L^{(j+1)} = L^{(j)} + \alpha L^{(j)}_{ext} \tag{5.2}$$

where $0 < \alpha \leq 1$ is a damping coefficient. The updated LLR $L^{(j+1)}$ is cyclically shifted by θ symbols, where θ is a random integer uniformly distributed between $(0, n-1)$. Since RS codes are cyclic, the cyclically shifted version of x is a valid codeword. Hence, a shifted version of $L^{(j+1)}$ can be thought of as the received signal when a shifted version of another valid codeword was transmitted. Therefore, another iteration of the SPA

is performed with the shifted version of the LLR $L^{(j+1)}$. Since the geometry of the factor graph associated with the shifted version is different from that of the original one, deterministic errors can be eliminated. We continue this procedure for a predetermined

Algorithm 1 SSID algorithm for RS codes

Step 1. Initialization: set $q = 0$,$j = 0$ and α_0.

Step 2. Set the coded bits LLR as observed from the channel: $L^{(0)}(x^i) = \frac{2}{\sigma^2} y_i$

Step 3. SPA: Feed the LLRs into the decoder and generate extrinsic LLRs for each bit using SPA: $L_{ext}^{(j)} = \psi(L^{(j)})$.

Step 4. Parameter Update: Update the LLR of each bit: $L^{(j+1)}(x^i) = L^{(j)}(x^i) + \alpha L_{ext}^{(j)}(x^i)$ where α is a gradually increasing damping coefficient to control the updating step width.

Step 5. Random Shifting: Cyclicly shift the LLRs by θ symbols and record the overall shift Θ: $L^{(j+1)} \longleftarrow L_{\theta}^{(j+1)}$

Step 6. Hard decision: $\widehat{c}_i = \begin{cases} 0, & L^{(j+1)}(x^i) > 0 \\ 1, & L^{(j+1)}(x^i) < 0 \end{cases}$

Step 7. Termination Criterion: If all the checks are satisfied, stop iteration and go to **Step 9**, else if $j = j_{max}$, go to **Step 8**, otherwise set $j \leftarrow j + 1$ and go to **Step 3** for another SPA iteration.

Step 8. Outer Round: If $q = q_{max}$ declare a decoding failure, otherwise set $q \leftarrow q + 1$ and $j = 0$,update the damping coefficient $\alpha = \alpha_0 + (q/(q_{max} - 1))(1 - \alpha_0)$ and go to **Step 2** for another outer round.

Step 9. Extract Information Bits: Shift the decoded bits back to their original position and get the information bits from coded bits. $\widehat{c} = \widehat{c}_{(-\Theta)}$

Table 5.1: Algorithm B

number of times or until the parity check equations are satisfied. When the maximum of j_{max} iterations is reached, another outer round, with a different realization of the random shifts and an increased α, begins with the original LLR from the channel, which prevents SPA decoding from getting stuck at pseudo-equilibrium points.

Define $\psi(L)$ as an one iteration of the SPA algorithm function with the input LLR L. Define L_θ as a cyclic shift of the vector by θ symbols (Note that received symbols should be shifted at symbol level). A detailed description of the algorithm is then given in Algorithm B.

However, there will has some problems as the codeword length becomes long. This is mainly due to the fact that the parity check matrix has high density and correlated unreliable information bits cause *error propagation*. Therefore, the above algorithm should be modified.

5.2 Iterative Decoding Algorithm By Adaptive Parity Check Matrix

Consider a narrow sense (N, K) RS code over GF(2^m), which has a parity check matrix H_s over GF(2^m). Let $n = N \times m$ and $k = K \times m$ be the length of the codeword and the number of information bits, respectively. H_s has an equivalent binary image expansion H_b, where H_b is an $(n-k) \times n$ binary check matrix.

Let $\underline{c} = [c_1, c_2, \cdots, c_n]$ be the binary representation of an RS codeword. Jiang and Narayanan (JN) [22] proposed an iterative BP-based soft-decision RS decoding algorithm that is composed of two stages: the matrix updating stage and the bit-reliability updating stage. In the matrix updating stage, the magnitude of the received LLR's $|L(c_i)|$ are first sorted so that $i_1, i_2, \cdots, i_{N-K}, \cdots, i_n$ denote the positions of the bits in terms of ascending order of $|L(c_i)|$, i.e., the bit c_{i_1} is the least reliable and c_{i_n} is the most reliable. We begin with the original parity check matrix H_b and perform elementary column operations to convert the i_1th column of H_b into the form $[1\ 0\ \cdots\ 0]^T$. Then, we convert the i_2th column of H_b into the form $[0\ 1\ 0\ \cdots\ 0]^T$ and so on. Finally, we transform $(n-k)$ columns among the n columns of H_b into the identity matrix. In the second stage, a standard BP algorithm is invoked to update the bit reliabilities.

The JN algorithm can be summarized as follows.

The JN Algorithm :

Initialization: Given the vector $\mathbf{\Lambda^{in}}$ of initial LLRs, the BP algorithm outputs the extrinsic LLR's $\mathbf{\Lambda^{x}}$ and let $\mathbf{\Lambda^{p}}$:=$\mathbf{\Lambda^{ch}}$

Do

1. Sort $\mathbf{\Lambda^{p}}$ in ascending order of magnitude and store the sorting index. The resulting vector of sorted LLRs is

$$\mathbf{\Lambda^{in}} = [\Lambda_1^{in}, \Lambda_2^{in}, \cdots, \Lambda_n^{in}]$$

   $||\Lambda_k^{in}||_1 \leq ||\Lambda_{k+1}^{in}||_1$ for $k = 1, 2, \cdots, n-1$ and $\mathbf{\Lambda^{in}} = P\mathbf{\Lambda^{p}}$, where P defines a permutation matrix.

2. Rearrange the columns of the binary parity-check matrix H_b to form a new matrix H_P, where the rearrangement is defined by the permutation P.

3. Perform Gaussian elimination(GE) on the matrix H_P from left to right. GE will reduce the first independent $(n-k)$ columns in H_P to an identity submatrix. Let this new matrix be $\widehat{H}_P$.

4. Run log BP on the parity-check matrix $\widehat{H}_P$ with initial LLR's $\mathbf{\Lambda^{in}}$ for a maximum number of iterations I_{t_H}. The log BP algorithm outputs extrinsic LLR's $\mathbf{\Lambda^{x}}$.

5. Update the LLR's via

$$\mathbf{\Lambda^{q}} = \mathbf{\Lambda^{in}} + \alpha_1 \mathbf{\Lambda^{x}}$$

   and

$$\mathbf{\Lambda^{p}} := P^{-1}\mathbf{\Lambda^{q}}$$

   where $0 < \alpha_1 \leq 1$ is called the ABP damping factor and P^{-1} is the inverse of P.

6. Make bit hard-decision based on $\mathbf{\Lambda^{p}}$ and use the bit-level parity check matrix to check if it is a valid codeword. If it passes the parity check then stop.

While the number of iterations $<$ the maximum number of iterations N_1

Jiang and Narayanan [22] proposed running N_2 outer iterations, each with the JN stopping criterion and a maximum of N_1 inner iterations. Each one of these N_2 iterations starts with a different random permutation of the sorted channel LLRs in the first inner iteration. We also run N_2 outer iterations, each with the list-decoding stopping criterion, to form a global list of at most N_1N_2 codewords. Finally, from the N_1N_2 codewords, choose the most probable one.

El-Khamy and McEliece (EM) [23] modify the JN algorithm by sending $\mathbf{\Lambda^P}$ to a KV decoder. They showed that the significant performance gain can be obtained by using the additional KV decoder.

5.3 Iterative Decoding of Serial Concatenated RS-Convolutional Codes

Based on the iterative SISO bit-level RS decoder of Section 5.1, we propose an architecture of serial concatenated RS and convolutional code with SISO iterative decoding. A block diagram of the proposed decoding scheme is depicted in Fig. 5.1. The convolutional code's constrain length is 7 and the code rate is 1/2. The BCJR algorithm is used for MAP decoding. A SOVA or soft-output BCJR decoder is used to obtain soft output

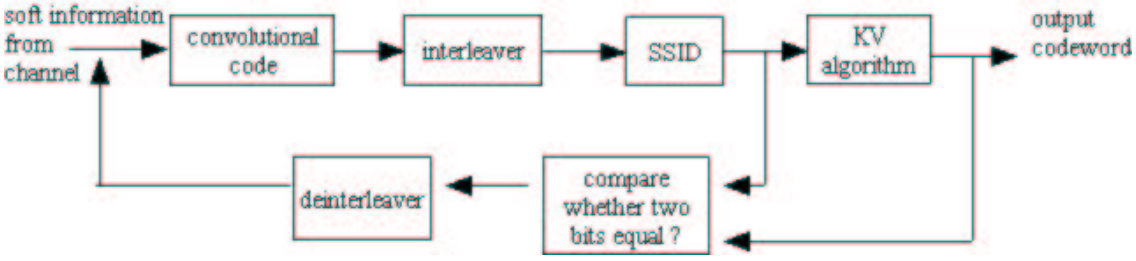


Figure 5.1: Soft-input soft-output iterative serial concatenated convolutional-RS codes

from the convolutional decoder. The soft output is passed to the SSID whose output is used as the channel reliability for the KV decoder. The KV decoder's HD codeword is

compared with the SSID soft output to see whether the two decoded bits are equal. If the two bits coincided, we feedback the soft bit level information of log-likelihood ratio p_1/p_0 from SSID to the convolutional decoder input as the a prior information. If the two bits are not the same, we set the log-likelihood ratio $p_1/p_0 = 0(p_1 = p_0)$. This is our SISO serial concatenated decoding algorithm.

Based on the discussion presented in Section 5.2, we have the modified concatenated code decoding algorithm given by the block diagram of Fig. 5.2 where the most probable codeword is chosen and the most reliable LLRs are sent back to the CC decoder input.

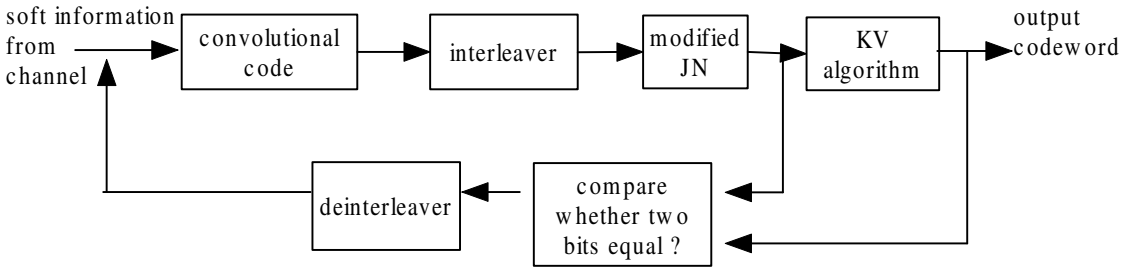


Figure 5.2: Iterative decoder structure for serial concatenated convolutional-RS codes

Chapter 6

Numerical Results

In this chapter, we present numerical results obtained by computer simulations of various decoding algorithms discussed so far.

Figs. 6.1 and 6.2, we can see that in the fixed codeword length n, as the k decrease (the code-rate decrease), there will be more coding gain.

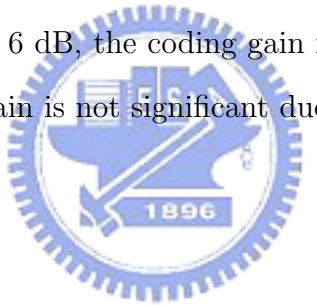
Consider the (31,25) RS code with BPSK modulation over AWGN channel. The performance of several schedules are shown in Fig. 6.2. We note that the SSID with BM algorithm with SPA=10, outer=5 and I_{th}=2 outperforms classical hard-decision decoding by 0.2dB at an FER of 10^{-3}. We can expect that the performance of SSID with BM can be better, if the SPA and the outer rounds are larger. However, in this way the complexity is quite high. It is a trade-off between complexity and error correcting capability. For the KV algorithm, we simulate it based on the simplified reencoding KV algorithm with $m_{\max} = 4$. Therefore, the interpolation cost is at most 60 iterations. It is a reasonable scheme. We can see that the KV algorithm outperforms the classical hard-decision decoding by 0.7 dB at an FER of 10^{-4}.

In Figs. 6.3 and 6.4, we show the most used case in different codeword length. Fig. 6.4 shows the performance of the high rate (0.9373) (255,239) RS code with BPSK modulation over an AWGN channel. Due to the high code rate, we can see that the performance gain is not impressive. The KV algorithm with $m_{\max} = 4$ outperforms the classical hard-decision decoding by 0.2 dB at an FER of 10^{-4}. The interpolation cost is

at most 160 iterations.

Fig. 6.5 presents the BER performance of serially-concatenated (255,239) RS-(7,1/2) convolutional code using the KV algorithm with 16-QAM modulation over an AWGN channel. The decoding algorithm for CC is the soft-input soft-output (SISO) BCJR algorithm. The soft CC and KV RS decoder has significant gain about 3dB at an BER of 10^{-4} over the classical all hard-decision RS-CC code.

In Fig. 6.6, we show the BER performance of the SISO iterative decoder for the serial concatenated code. We use the decoder shown in Fig. 5.1. The SSID algorithm uses SPA=10, outer rounds=5, I_{th}=2. The soft CC and SSID with KV($\mathrm{m_{max}}$)=4 with pass 1 outperforms the HD decoder by about 3.1 dB at BER = 10^{-5}, with pass 2 by a margin of 3.3 dB, and with pass 3 by about 3.4 dB. We also simulate the SISO algorithm shown in Fig. 5.2. Due to the required extreme long simulation time, we simplify the case with N_1=5 (inner iterations), N_2=2 (outer iterations), and I_{th}=2. We find a performance gain of 5 dB while at the SNR = 6 dB, the coding gain is about 4 dB at BER = 10^{-5}. However the iterative decoding gain is not significant due to the small N_1 and N_2.



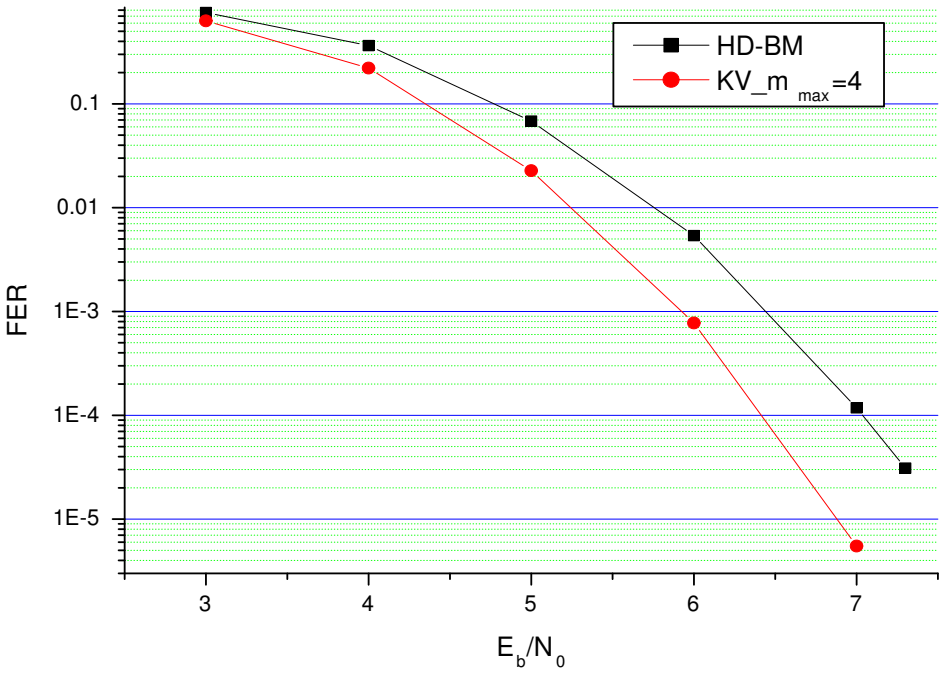


Figure 6.1: (31,23) RS code with BPSK modulation over AWGN channels; code rate=0.7419

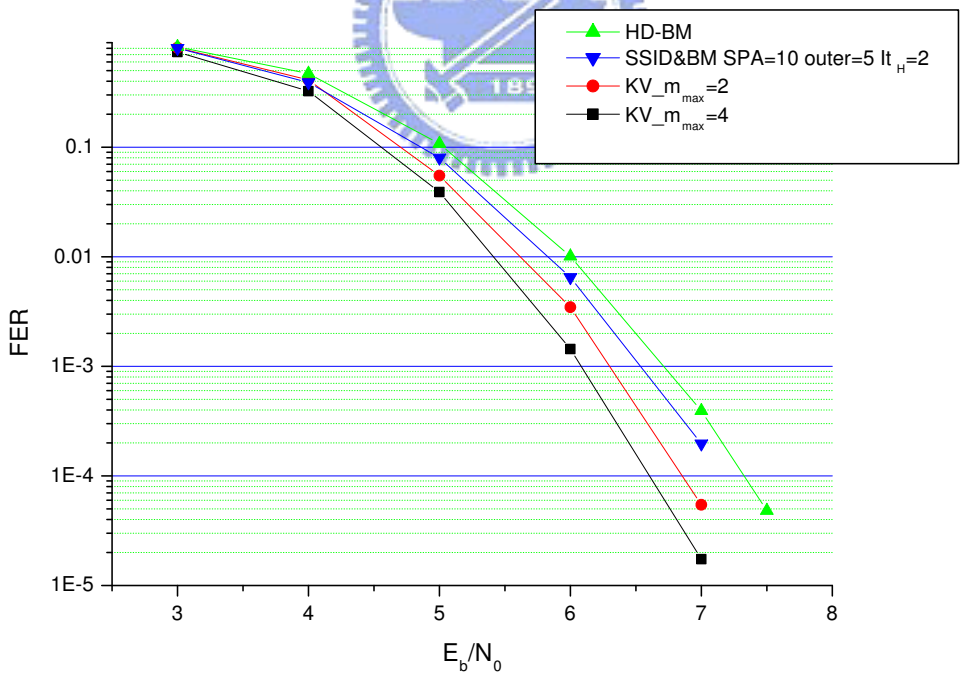


Figure 6.2: (31,25) RS code with BPSK modulation over AWGN channel, code rate=0.8065

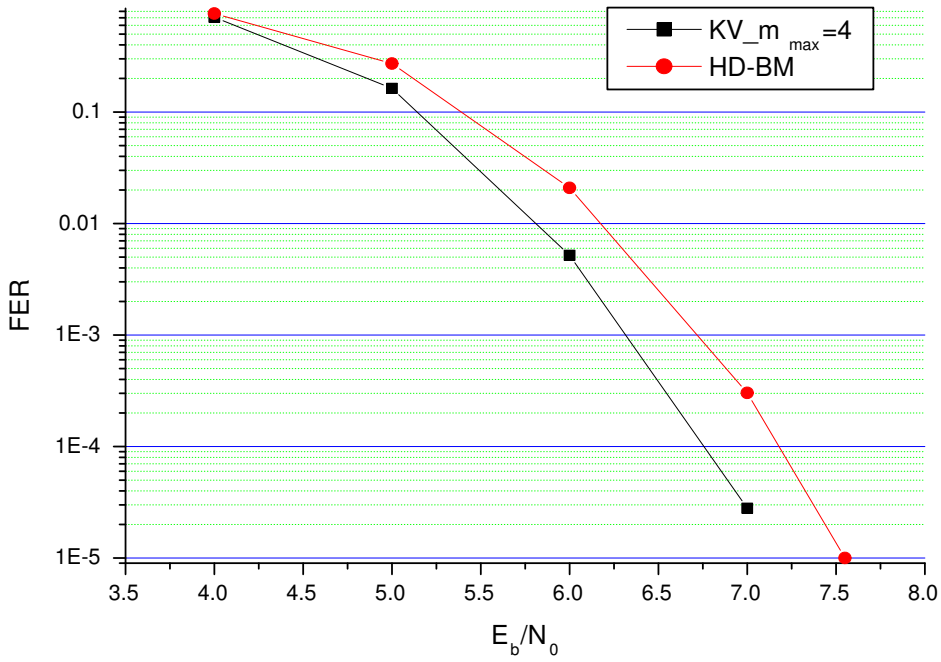


Figure 6.3: (63,55) RS code with BPSK modulation over AWGN channels; code rate=0.8730

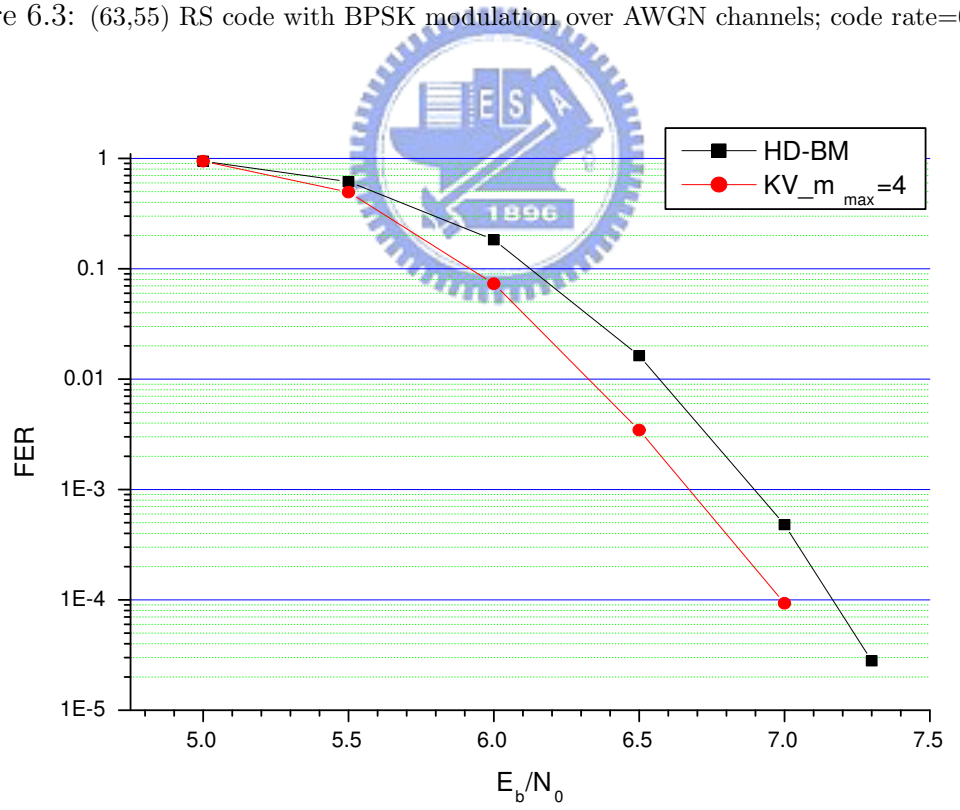


Figure 6.4: (255,239) RS code with BPSK modulation over AWGN channels; code rate=0.9373

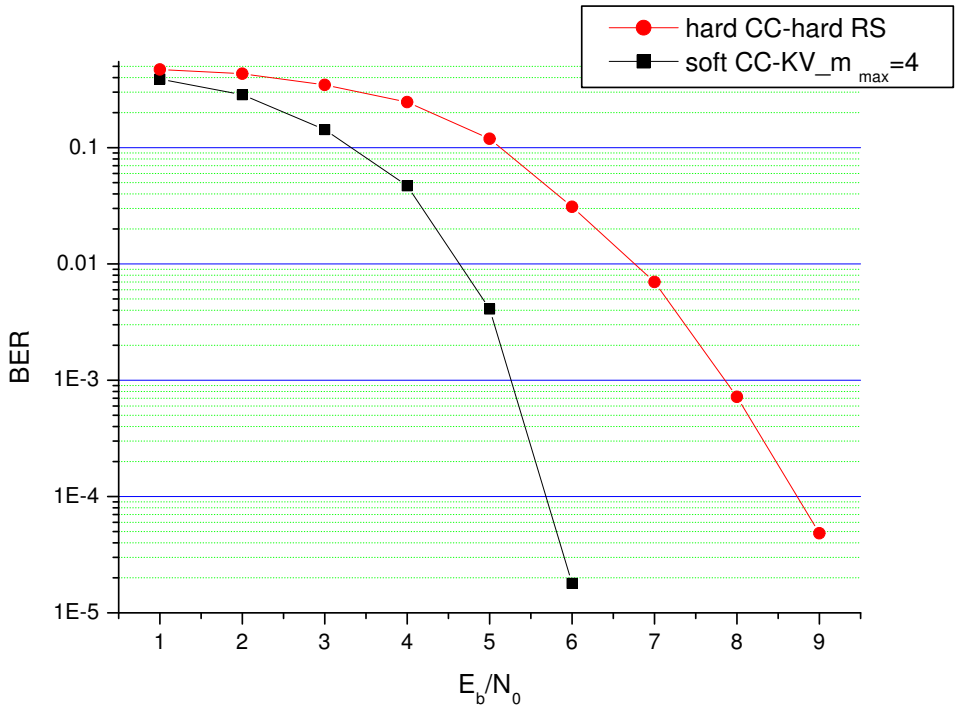


Figure 6.5: Serial concatenated (255,239) RS and (7,1/2) CC code with 16-QAM modulation over AWGN channels; code rate=0.4668

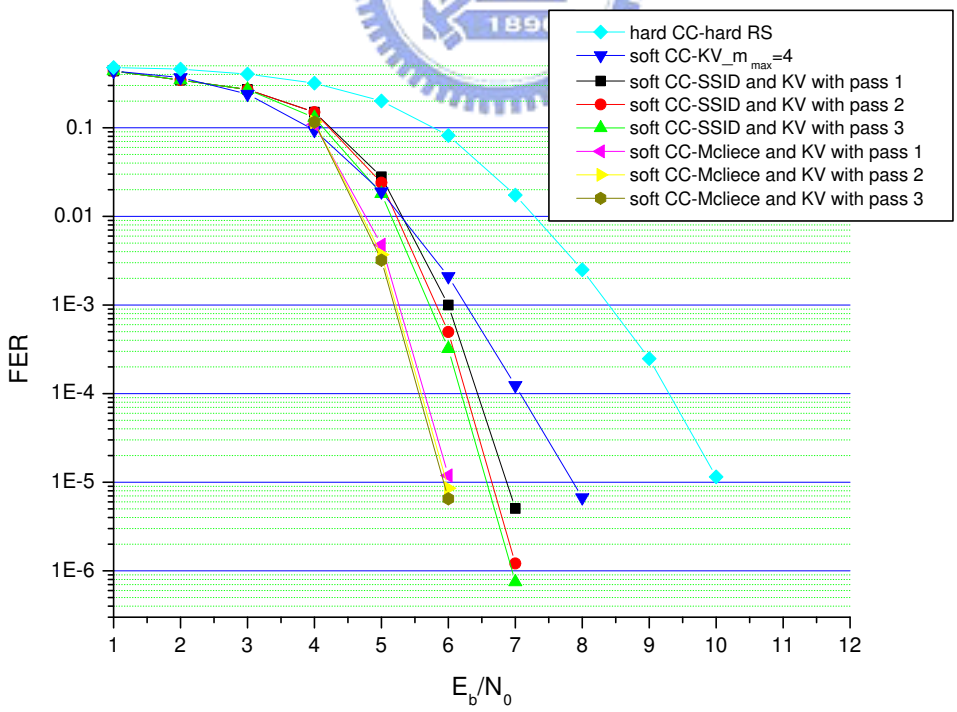


Figure 6.6: Serial concatenated (31,25) RS and (7,1/2) CC code with 16-QAM modulation over AWGN channel, code rate=0.4017

Chapter 7

Conclusion

In this thesis we investigate the performance of several coding systems involved RS codes. We first examine the performance of SD decoded RS codes and serial concatenated RS-CC codes using modified KV algorithm, using a transformation of the received word to reduce the iterative decoding complexity.

We then study the effectiveness of using the SSID algorithm in conjunction with the KV algorithm for iterative decoding of serial concatenated codes. An alternate iterative decoding structure replaces the SSID algorithm by the EM algorithm which eliminates short cycles in long RS codes and gives improved SD decoding performance.

Performance gain of SD RS decoder and iterative serial concatenated decoder is evaluated through simulations. But the complexity is still significant high using the JN algorithm to derive a soft output associated with a RS code. A reliable and easily-generated soft RS decoder output is highly desired for it will make iterative decoding of serial concatenated RS-CC codes practically realizable and the applications of such a practical decoder are too numerous.

Bibliography

[1] M. Sudan, "Decoding of Reed-Solomon Codes beyond the Error-Correction Bound," *J. Complexity*, vol. 13, pp. 180-193, 1997.

[2] V. Guruswami and M. Sudan, "Improved Decoding of Reed-Solomon Codes and Algebraic Geometry Codes," *IEEE Trans. Inform. Theory*, vol. 45, no. 6, pp. 1757-1767, Sep. 1999.

[3] R. J. McEliece, "The Guruswami-Sudan decoding algorithm for Reed-Solomon codes," *IPN Progress Report 42-153*, May 15. 2003.

[4] R. Roth and G. Ruckenstein, "Efficient Decoding of Reed-Solomon Codes beyond Half the Minimum Distance," *IEEE Trans. Inform. Theory, vol. 46, no. 1, pp. 246-257,* Jan. 2000.

[5] W. J. Gross, F. R. Kschischang, R. KÖtter, and P. G. Gulak, "Towards a VLSI Architecture for Interpolation-Based Soft-Decision Reed-Solomon Decoders," *J. VLSI Signal Proc.*, Jul. 2003

[6] S. Wicker, *Error control systems for Digital communication and storage*, Prentice Hall, USA, 1995.

[7] R. KÖtter, J. Ma, A. Vardy, and A. Ahmed, "Efficient interpolation and factorization in algebraic soft-decision decoding of Reed-Solomon codes." *Submitted to ISIT 2003.*

[8] R. K0tter and A. Vardy, "A complexity reducing transformation in algebraic list decoding of Reed-Solomon codes," in *Proc. of ITW2003*, Paris, France, March 31-April 4 2003.

[9] R. K0tter, and A. Vardy, "Algebraic Soft-Decision Decoding of Reed-Solomon Codes," *IEEE Trans. Inform. Theory*, vol.49, Nov. 2003.

[10] W. J. Gross, F. R. Kschischang, R. K0tter, and P. Gulak, "A VLSI architecture for interpolation in soft-decision list decoding of Reed-Solomon codes," in *Proc. 2002 IEEE Workshop Signal Process. Sys.* (SIPS02), pp. 39-44, San Diego, CA, USA, Oct. 16-18 2002.

[11] G. D. Forney, Jr., *Concatenated codes*, MIT Press, Cambridge, MA, USA, 1966.

[12] G. D. Forney, Jr., "Generalized minimum distance decoding," *IEEE Trans. Inform. Theory*, vol. IT-12, pp. 125-131, Apr. 1966.

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

[14] A. Ahmed, R. K0tter, and N. R. Shanbhag, "VLSI architectures for soft-decision decoding of Reed-Solomon codes," *Submitted to IEEE Transactions on VLSI Systems,* Feb. 2003.

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

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

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

[18] Y. Wu, R. KÖtter, and C. Hadjicostis, "Soft-decision decoding of linear block codes using preprocessing," in *Proc. ISIT*, Chicago, IL, USA, Jul. 2004.

[19] T. Hu and S. Lin, "An efficient hybrid decoding algorithm for Reed- Solomon codes based on bit reliability," *IEEE Trans. Commun.*, vol. 51, pp. 1073-1081, Jul. 2003.

[20] M. P. C. Fossorier and A. Valembois, "Reliability-based decoding of Reed-Solomon codes using their binary image," *IEEE Commun. Letr.*, vol. 7, pp. 452-454, Jul. 2004.

[21] J. Jiang, and K. R. Narayanan, "Iterative Soft Decoding of Reed-Solomon Codes," *IEEE Commun. Lett.,vol.8* Apr. 2004

[22] J. Jiang and K. Narayanan, "Iterative Soft Decision Decoding of Reed Solomon Codes Based on Adaptive Parity-Check Matrices," *in Proc. Int. Symp. Inf. Theory,* p.261, Jul. 2004.

[23] M. El-Khamy, and R. J.Mcliece "Iterative Algebraic Soft-Decision List Decoding of Reed-Solomon Codes," *IEEE Journal on selected areas in com., vol.24 NO 3.,* March 2006

作 者 簡 歷

楊哲雄，高雄縣人，1981 年生

私立瀛海高級中學	1997.9 ~ 2000.6
國立交通大學電子工程學系	2000.9 ~ 2004.6
國立交通大學電信工程學系系統組	2004.9 ~ 2006.6

Graduate Course:

1. Coding Theory
2. Random Process
3. Special Topics in Digital Signal Processing
4. Detection and Estimation Theory
5. Special Topics in Communication Systems
6. Adaptive Signal Processing
7. Digital Signal Processing
8. Queuing Theory