December 11, 1998

To the Graduate School:

This dissertation entitled "Soft-Decision Decoding of Nonbinary Cyclic and Modified Cyclic Codes" and written by Laurie L. Joiner is presented to the Graduate School of Clemson University. I recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy with a major in Electrical Engineering.

John J. Komo, Dissertation Advisor

We have reviewed this dissertation
and recommend its acceptance:

Carl W. Baum

Joel V. Brawley, Jr.

Michael B. Pursley

Accepted for the Graduate School:

SOFT-DECISION DECODING OF NONBINARY CYCLIC AND MODIFIED CYCLIC CODES

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Electrical Engineering

---

by
Laurie L. Joiner

December 1998

Advisor: Dr. John J. Komo

ABSTRACT

Cyclic codes and modified cyclic codes are widely used because they can be decoded efficiently. These codes are usually decoded with hard-decision decoders that take advantage of the algebraic properties of the codes. In this dissertation algorithms are presented which use the algebraic structure of cyclic codes to provide efficient soft decision decoding of Reed-Solomon codes. These algorithms provide better error correction performance than existing hard-decision decoding algorithms, and more efficient decoding than existing soft decision decoding techniques.

Reed-Solomon codes are powerful codes, but they are limited to certain code lengths. For codes with lengths that cannot be Reed-Solomon lengths, nonbinary BCH codes are used. These codes can be decoded using the same hard-decision decoding techniques that are used with Reed-Solomon codes, but can only decode up to their design distance. The design distance of a nonbinary BCH code is generally less than the minimum distance of a cyclic code of the same dimensions. Cyclic codes and shortened cyclic codes are presented which have better minimum distances than corresponding nonbinary BCH codes. These cyclic codes do not have efficient hard-decision decoding algorithms. However, efficient algorithms for soft-decision decoding of these codes are presented.

Normally, for error control in a communication system using M-ary orthogonal signaling, a Reed-Solomon code of length M is used. To achieve good error performance, a large value of M may be desired, but this results in a complex receiver system. A method is presented for using a smaller signal constellation size, n, while maintaining system performance. Systems are developed which have constant bandwidth and improve error performance while decreasing system complexity. These systems use Reed-Solomon and cyclic codes, and they use soft-decision decoding techniques.

DEDICATION

To Ron for all of his love and support. The next one is yours.

ACKNOWLEDGMENTS

I would like to thank my advisor, John Komo for his years of advice and encouragement. I would also like to thank my committee members, Carl Baum, Joel Brawley, and Michael Pursley. They have all given excellent advice and been wonderful instructors. I would also like to thank Ron Bowman for proofreading yet another thesis and for always convincing me that I could do this. Finally, I would like to thank my parents for instilling in me the importance of education and for always encouraging me to be my best.

TABLE OF CONTENTS

Table of Contents (Continued)

LIST OF TABLES

LIST OF FIGURES

List of Figures (Continued)

Figure Page

CHAPTER 1

INTRODUCTION

Error control codes add redundancy to a string of information symbols to enable a decoder to correct errors that occur during the transmission of information over a noisy channel. For a block code the encoder performs a one-to-one mapping of a string of k information symbols to a string of n code symbols. These n code symbols make up a codeword, $\mathbf{c}$. The information and code symbols are from a finite alphabet of q symbols, generally taken to be a Galois field of order q and denoted by GF(q). A code symbol, c_i, is transmitted over a channel which adds interference and produces an output $\mathbf{y}_i$, which is, for example, the set of voltages at the output of the matched filters. It is the decoder's job to determine, from n of these transmissions, which codeword was transmitted.

If the detector maps a received signal to one symbol of the signaling alphabet, the receiver has made a hard-decision. This hard-decision results in the decoder discarding all information about the reliability of the received symbol. The decoder performs hard decision decoding by treating all received symbols the same since it does not know which symbols are more reliable.

If the detector can choose from $q+1$ possible outputs, then one output is denoted as an erasure. An erasure is an output that is believed to be unreliable. The decoder can use this information to correct more errors than in normal hard-decision decoding and thus obtain better performance (lower probability of not decoding correctly).

To further increase performance, the detector could output a reliability vector, α_i, for each transmitted symbol. This reliability vector contains information about the confidence of the receiver's decisions. A decoder that utilizes the reliability vector is called a soft-decision decoder. The performance of a soft decision decoder is better than a hard-decision decoder; however the decoding system is more complex.

The two main types of error control codes are block codes and convolutional codes. Since block codes are developed with tools from abstract algebra, efficient hard-decision decoding algorithms exist. Convolutional codes do not use algebraic structure in decoding algorithms, but soft-decision decoding algorithms providing maximum-likelihood decoding exist for these codes. The focus of this work is on the development and performance of decoding algorithms for block codes that provide the error correcting capabilities of maximum-likelihood decoding using the algebraic properties of block codes.

One of the most popular families of block codes is the family of Reed-Solomon (RS) codes. These codes are nonbinary (q-ary) codes over GF(q) with length $n = q-1$. RS codes are maximum distance separable, which means that they satisfy the Singleton bound ($d_{\min} \leq n-k+1$) with equality. Thus, they have minimum distance of $d_{\min} = n-k+1$. There are several efficient hard-decision decoding algorithms for RS codes that take advantage of the algebraic properties of the code [1]. In addition, there are soft decision algorithms that do not use the algebraic properties of the codes [2]. New soft-decision decoding algorithms are presented in Chapter 2. These algorithms provide better performance than hard-decision decoding algorithms, and they are more efficient than existing soft-decision decoding algorithms.

One disadvantage of RS codes is the restriction on the length of the code to be approximately the same as the code alphabet size. To obtain a longer RS code, the code alphabet must be increased which may not be possible if q-ary code symbols are required. It is possible to concatenate two shorter RS codes that use the desired code symbol alphabet to get a longer code, but usually, longer codes should provide better error correction capabilities than shorter codes. Thus, q-ary codes are needed whose length is not dependent on the code alphabet size. One class of q-ary codes that can have longer lengths are the general class of cyclic codes. Cyclic codes can be found for almost any desired length. Unfortunately certain code lengths with large minimum distances cannot be found. However, it is possible to obtain codes of desired lengths by shortening longer codes, where the minimum distance of the shortened code will be at least as large as the minimum distance of the original cyclic code.

Within the class of cyclic codes are nonbinary BCH codes. The design process for these codes specifies a design distance, δ where $\delta \leq d_{\min}$. BCH codes are commonly used since efficient hard decision decoding algorithms exist for these codes. Unfortunately, these hard-decision algorithms only decode up to the design distance which can be less than the minimum distance for that code, and may be much smaller than the minimum distance for the same length non-BCH cyclic code.

The soft-decision algorithms derived in Chapter 2 can be modified to correct non-RS codes. These algorithms do not depend on the design distance of the code, so the larger class of cyclic codes are searched to find codes with the best possible minimum distance. Cyclic and shortened cyclic codes and decoding algorithms for these codes are presented in Chapter 3.

In Chapter 4 systems using M-ary frequency shift keying and error control codes are compared using the codes presented in Chapter 3 and the decoding algorithms developed in Chapter 2. Comparisons of the systems are made using constant bandwidth and power constraints. Hard and soft-decision decoding algorithms are used with these systems.

CHAPTER 2

SOFT-DECISION DECODING OF REED-SOLOMON CODES

An (n, k) Reed-Solomon code is a q-ary code of length n, dimension k,and minimum distance of $d_{\min} = n - k + 1$. In general if a code has minimum distance $d_{\min}$, then it can correct t errors, where $t \leq \lfloor (d_{\min} - 1)/2 \rfloor$. The codes are based on algebraic properties and have efficient decoding algorithms [1] for hard-decision decoding. The algorithms have two parts since the codes are non-binary codes. First, the decoder finds the error locations, and second, the decoder determines the magnitudes of the errors.

If a received symbol is believed to be unreliable, then it can be marked as an erasure. If a decoder can use the erasure information, then the decoder is called an errors and erasures decoder. When an erasure is given to the decoder, the error location is known. Therefore, the decoder determines only the magnitude of the error. In general a code can correct t errors and s erasures as long as $(2t + s) < d_{\min}$. Thus, if a receiver can correctly erase the symbols that are in error, twice as many errors can be corrected in errors and erasures decoding than with errors only decoding. The algorithms that exist for errors only decoding of RS codes are easily modified to efficiently perform errors and erasures decoding [1].

While an errors and erasures decoding algorithm uses some of the reliability information from the detector, it still does not use all of the available information. One method of using the information available at the output of the channel is to let the detector determine a reliability for each possible symbol based on the received signal. The decoder uses these reliabilities to determine the codeword that has the highest probability of matching the transmitted codeword. This approach generates the codeword that has the maximum likelihood of being sent given the received word. Such decoding results in low probability of decoder error, but current implementations have a high cost of decoder complexity and decoding time. In this chapter algorithms are presented which efficiently perform maximum-likelihood or near maximum-likelihood decoding by using algebraic decoding algorithms.

2.1 Decoding Algorithms

One method of decoding RS codes uses the reliability information to determine which symbols of the received word are most likely in error, and to erase the maximum number of these symbols, $n-k$. This method allows a decoder to correct up to twice the number of errors of the errors only decoding approach. In addition, if the error locations are known a decoder finds error magnitudes efficiently. The problem with this approach is errors which lie outside of the $n-k$ erasures are undetected and thus, uncorrected. To solve this problem methods of looking at more than one set of errors and choosing the resulting codeword with the highest probability of being the transmitted codeword are presented.

2.1.1 Sequential Decoding of Reed-Solomon Codes

Let $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$ be the transmitted codeword. The code symbol, c_i is transmitted across a noisy channel. The output from the receiver is a vector, $\mathbf{y}_i$ which is, for example, the set of output voltages of a receiver's matched filters. This vector is processed into a hard-decision estimate, r_i, and a reliability vector, $\mathbf{m}_i = (m_{i,0}, m_{i,1}, \ldots, m_{i,q-1})$. The j-th element of $\mathbf{m}_i$ denotes how probable the symbol transmitted at the i-th location was the j-th symbol. Thus, if $m_{i_j} > m_{i_{j'}}$ it is more likely that symbol j was transmitted than symbol j' at location i. The reliability vector is reordered so that the most reliable element is the first element in the vector; denote the sorted vector as $\mathbf{m'}_i = (m'_{i,0}, m'_{i,1}, \ldots, m'_{i,q-1})$ which corresponds to symbol values $\mathbf{z} = (z_{i,0}, z_{i,1}, \ldots, z_{i,q-1})$, where $z_{i,0}$ is the hard-decision estimate of $\mathbf{y}_i$, i. e., $z_{i,0} = r_i$.

The metrics are normalized by subtracting $m'_{i,0}$ from each component of $\mathbf{m'}_i$ so the second component of $\mathbf{m'}_i$, $m'_{i,1}$ is an indication of the overall reliability of the symbol decision for the i-th location. A large magnitude value for $m'_{i,1}$ implies the hard-decision choice, r_i, is a reliable choice. After a codeword has been transmitted, the n reliability vectors are used to sort the symbol locations in order of reliability. Let β_i denote the location of the symbol with the i-th largest metric, $m'_{i,1}$. The values of β_i denote the reliability ordering and satisfy $|m'_{\beta_1,1}| > |m'_{\beta_2,1}| > \cdots > |m'_{\beta_n,1}|$. Since RS codes are maximum distance separable (MDS), any k locations in the codeword can be taken as the data locations for encoding. The k locations in the received word with the largest metrics, $\beta_1, \beta_2, \ldots, \beta_k$, are taken as the data locations for re-encoding. Equivalently, the $n-k$ locations with the smallest metrics, $\beta_{k+1}, \beta_{k+2}, \ldots, \beta_n$, are taken as erasure locations for erasures-only decoding.

A stack is created using symbol values for a sequence of data locations which correspond to the metrics $m'_{\beta_1,0}, m'_{\beta_2,0}, \ldots, m'_{\beta_k,0}$. After this stack is created, a new metric is calculated which is the sum of the metrics of the symbols in the codeword, including the parity symbols. If this metric is larger than any metric currently in the stack, the decoding is complete. Otherwise, the stack is extended until the top of the stack corresponds to an entry with all of the data locations. Again, a new metric is calculated which includes the metrics of the corresponding parity symbols. The search for a near maximum-likelihood estimate of the codeword is complete when a codeword has the largest possible metric (a codeword is at the top of the stack). A flow chart for the algorithm is shown in Figure 2.1.

The results of this decoding method produce a near maximum-likelihood estimate of the transmitted codeword. Since the stack is begun with the k most reliable symbols, the sequential search is reduced considerably. The decoding that takes place with each iteration of the algorithm is erasures only decoding. Erasures only decoding can be performed quickly since only the error magnitudes and not the error locations must be found. A fast algorithm for performing the error magnitude calculations has been developed in [3].

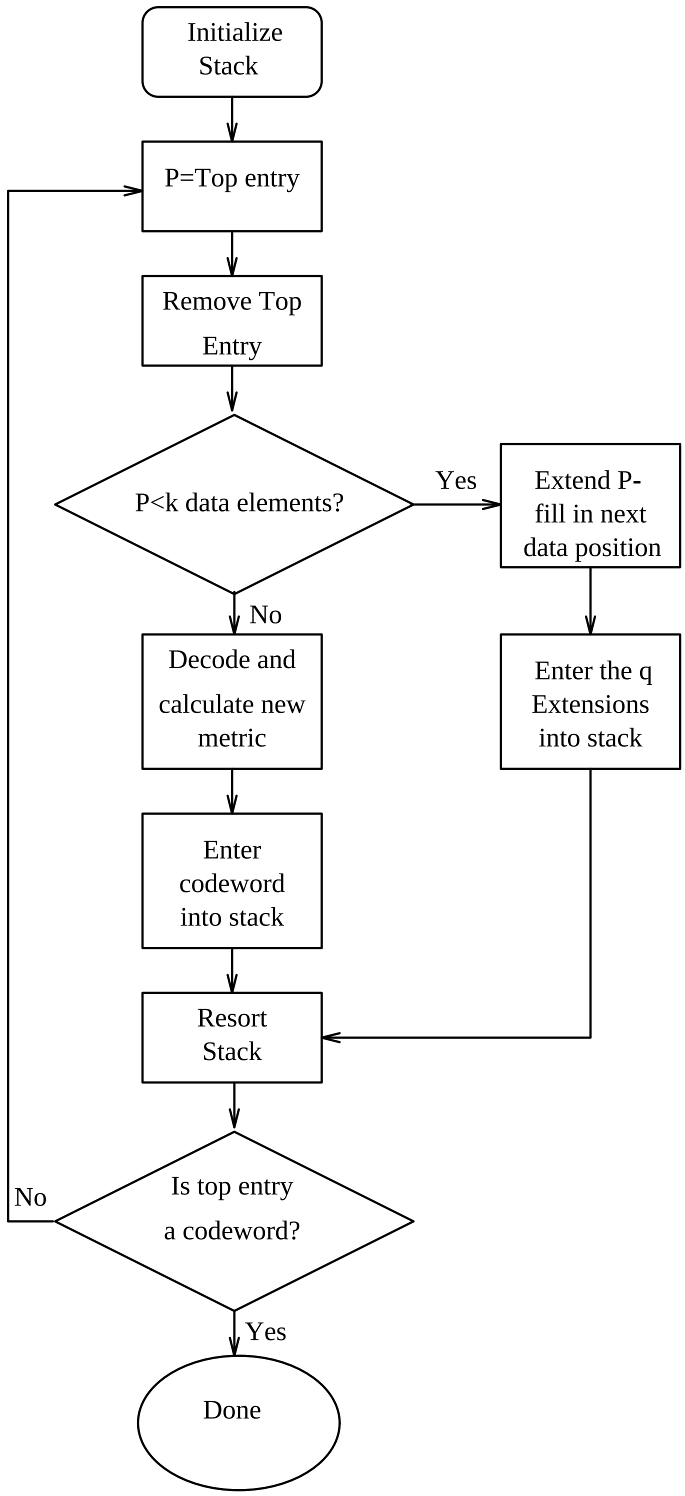

Figure 2.1 Flow Chart for Sequential Decoding Algorithm

Example 1: Suppose an $(8,4)$ extended RS code is transmitted over a Rayleigh fading channel using 8-ary modulation. The following table lists the metric for each of the eight symbols at the eight transmitted locations. The metrics are calculated for each transmitted code symbol from the outputs of the matched filters. Let $\mathbf{y}_i = (y_{i,0}, y_{i,1}, \ldots, y_{i,q-1})$ be the output voltages for the i-th transmitted symbol. The metrics are found by taking the differences of the ordered vector outputs; $\mathbf{m}'_i = (0, y'_{i,1} - y_{i,0}, \ldots, y'_{i,q-1} - y_{i,0})$, where $y'_{i,0} = \max(y_{i,0}, y_{i,1}, \ldots, y_{i,q-1})$. To find the actual symbol reliability requires knowledge of the symbol energy to noise ratio. However, an estimate of symbol reliability can be obtained from the differences of the output voltages. With this estimation a large difference between the largest filter output and the next largest filter output for any symbol location is assumed to be a more reliable symbol decision and results in an more negative metric value for the second most likely symbol. For this example, location 4 is assumed to be the most reliable symbol (symbol 0) since it has the largest (most negative) second metric value (-41.43).

Location 0		Location 1	
Symbol Value	Metric Value	Symbol Value	Metric Value
0	0.00	0	0.00
α^5	-8.70	α^2	-31.38
α^4	-9.03	α^6	-31.49
α^3	-9.14	α	-31.68
α^2	-9.24	1	-33.05
α	-9.88	α^5	-34.01
1	-9.98	α^3	-34.01
α^6	-10.07	α^4	-34.19
Location 2		Location 3	
Symbol Value	Metric Value	Symbol Value	Metric Value
0	0.00	0	0.00
α	-0.51	α	-1.77
α^4	-1.27	α^6	-2.10
1	-1.44	1	-2.23
α^3	-1.46	α^4	-3.22
α^6	-1.64	α^2	-3.46
α^2	-1.82	α^3	-3.51
α^5	-2.28	α^5	-3.54

Location 4		Location 5	
Symbol Value	Metric Value	Symbol Value	Metric Value
0	0.00	α^3	0.00
α^5	-41.43	α	-1.13
α^3	-42.27	α^5	-1.64
1	-42.34	α^4	-1.91
α	-42.48	0	-2.08
α^6	-42.72	α^6	-2.11
α^2	-42.82	1	-2.82
α^4	-42.83	α^2	-3.48
Location 6		**Location 7**	
Symbol Value	Metric Value	Symbol Value	Metric Value
α	0.00	0	0.00
α^4	-1.23	α^6	-4.52
1	-2.60	α^3	-5.37
α^5	-2.63	1	-6.11
α^3	-3.44	α^2	-6.11
α^6	-3.44	α^5	-6.53
α^2	-3.61	α^4	-6.62
0	-3.64	α	-7.05

The data locations are chosen to be the four positions which have the largest magnitude for the second metrics. Thus, from the above table, the data locations are $\beta_1 = 4$, $\beta_2 = 1$, $\beta_3 = 0$, and $\beta_4 = 7$. The positions that are left, $\beta_5 = 3$, $\beta_6 = 6$, $\beta_7 = 5$, and $\beta_8 = 2$ are assumed to be the less reliable positions and are erased in the decoder.

The stack is created by inserting symbol values for the first data position (here, location 4). The top entry is extended by filling in values for the second data position, location 1. The process continues until the top entry in the stack has symbol values for all data locations. At the point where the top entry in the stack is of length four, the stack (the first nine entries) is as follows:

Metric Value	$\beta_1 = 4$	$\beta_2 = 1$	$\beta_3 = 0$	$\beta_4 = 7$	$\beta_5 = 3$	$\beta_6 = 6$	$\beta_7 = 5$	$\beta_8 = 2$
0.00	0	0	0	0				
-4.52	0	0	0	α^6				
-5.37	0	0	0	α^3				
-6.11	0	0	0	1				
-6.11	0	0	0	α^2				
-6.53	0	0	0	α^5				
-6.63	0	0	0	α^4				
-7.05	0	0	0	α				
-8.70	0	0	α^5					

The top entry in the stack has all of its data locations filled, so the data sequence is decoded and the metric for the codeword is calculated. No stack entries with metrics smaller than the codeword metric will be considered, so the stack is shortened to

Metric Value	$\beta_1 = 4$	$\beta_2 = 1$	$\beta_3 = 0$	$\beta_4 = 7$	$\beta_5 = 3$	$\beta_6 = 6$	$\beta_7 = 5$	$\beta_8 = 2$
-4.52	0	0	0	α^6				
-5.37	0	0	0	α^3				
-5.72	0	0	0	0	0	0	0	0

The top two entries are decoded and their new metrics are calculated to be -13.08 and -13.81, respectively. After these decodings the stack becomes

Metric Value	$\beta_1 = 4$	$\beta_2 = 1$	$\beta_3 = 0$	$\beta_4 = 7$	$\beta_5 = 3$	$\beta_6 = 6$	$\beta_7 = 5$	$\beta_8 = 2$
-5.72	0	0	0	0	0	0	0	0
-13.08	0	0	0	α^6	α^2	α	1	α^5
-13.81	0	0	0	α^3	α^6	α^5	α^4	α^2

Since the top entry in the stack is a codeword, the decoding is complete and the decoded codeword is the all-zero codeword.

For large codes and for channels with low energy to noise ratios, the decoding time for a received word can be unfeasibly long because of the number of possible codewords that must be searched. The decoding time can be shortened in two ways. First the stack can be shortened by eliminating some of the least likely entries in the stack. By shortening the stack some of the codeword possibilities are eliminated and the number of iterations that must be performed is reduced. The resulting codeword is no longer a maximum-likelihood estimate but a near maximum-likelihood estimate.

Shortening the stack can also be viewed as limiting the number of decodings to a fixed number. By limiting the number of decodings, the stack size is also limited since there is no need to have a stack containing sequences that will not be extended. If the algorithm is limited to a fixed number of decodings, then a procedure must be developed to handle the situation of making the maximum number of decodings but not finding a codeword with the largest possible metric. If a system can handle decoder failures (such as being able to ask for retransmissions), then the decoder can declare a decoder failure. Some systems, such as a deep space telemetry system, cannot handle decoder failures. Thus, the decoder can either mark the codeword as unreliable and let a user determine how to interpret the data, or make a best guess as to the correct codeword (i. e. choose the codeword with the largest metric found so far). For the results shown in this dissertation, the latter option is chosen.

The decoding time can also be shortened by adjusting the metric to give an advantage to the longer paths that have been explored, similar to the Fano metric described in [4]. With the original decoding method, a sequence with few symbols can be expanded if it has a metric slightly larger than the metric of a longer sequence. Addition of a bias to longer length sequences, allows the longer sequences to be chosen over smaller length sequences of approximately the same metric value. The number of searches is reduced, but a sequence may be selected that does not have the largest possible metric. Near maximum-likelihood performance is obtained by using this method.

Example 2: In the previous example if a bias is added to the metrics, fewer decodings are performed. For example, adding a bias equal to the code rate of 0.5 to each metric, yields the same stack entries except for the metrics.

Metric Value	$\beta_1 = 4$	$\beta_2 = 1$	$\beta_3 = 0$	$\beta_4 = 7$	$\beta_5 = 3$	$\beta_6 = 6$	$\beta_7 = 5$	$\beta_8 = 2$
2.00	0	0	0	0				
-2.52	0	0	0	α^6				
-3.37	0	0	0	α^3				
-4.11	0	0	0	1				
-4.11	0	0	0	α^2				
-4.53	0	0	0	α^5				
-4.63	0	0	0	α^4				
-5.05	0	0	0	α				
-7.02	0	0	α^5					
-8.53	0	0	α^4					

Now, the metric for the all-zero codeword is calculated as -1.72 and the stack is

Metric Value	$\beta_1 = 4$	$\beta_2 = 1$	$\beta_3 = 0$	$\beta_4 = 7$	$\beta_5 = 3$	$\beta_6 = 6$	$\beta_7 = 5$	$\beta_8 = 2$
-1.72	0	0	0	0	0	0	0	0
-2.52	0	0	0	α^6				
-3.37	0	0	0	α^3				

This metric is the largest metric in the stack, so the decoding process is complete. Therefore, addition of the bias to the metrics, reduces the number of decodings for this example from three to one.

2.1.2 Bucket Algorithm

A disadvantage of the sequential algorithm presented in Section 2.1.1, is the sorting procedure that is performed after every extension or decoding. With a large stack size the sorting procedure is a time consuming process. The bucket algorithm eliminates the sorting procedure, but may increase the number of codewords considered since a non-optimal entry may be extended.

In the bucket algorithm the stack is divided into sections called buckets, and each bucket corresponds to a range of metrics. After a data sequence is extended, it is placed in the bucket corresponding to the entry's metric. The entries are not sorted within each bucket, and they are used as a last in first out stack. The codeword with the largest metric is stored separately and only entries with better metrics than the codeword metric are extended. The decoding process is complete when all buckets that could have better metrics than the metric of the codeword are empty.

Example 3: Using the data from the first example and buckets divided so that the i-th bucket contains the metrics from $-i$ to $-i+1$, the first three non-empty buckets are

Bucket 1								
Metric Value	$\beta_1 = 4$	$\beta_2 = 1$	$\beta_3 = 0$	$\beta_4 = 7$	$\beta_5 = 3$	$\beta_6 = 6$	$\beta_7 = 5$	$\beta_8 = 2$
0.00	0	0	0	0				

Bucket 5								
Metric Value	$\beta_1 = 4$	$\beta_2 = 1$	$\beta_3 = 0$	$\beta_4 = 7$	$\beta_5 = 3$	$\beta_6 = 6$	$\beta_7 = 5$	$\beta_8 = 2$
-4.52	0	0	0	α^6				

Bucket 6								
Metric Value	$\beta_1 = 4$	$\beta_2 = 1$	$\beta_3 = 0$	$\beta_4 = 7$	$\beta_5 = 3$	$\beta_6 = 6$	$\beta_7 = 5$	$\beta_8 = 2$
-5.37	0	0	0	α^3				

After the entry in Bucket 1 has been decoded, it has a metric of -5.72 and is placed in Bucket 6. The entries of Buckets 5 and 6 are then decoded and stored in Bucket 14 due to the codeword metrics of -13.08 and -13.81. The top six buckets are empty which are the only buckets that can have metrics better than the codeword metric, so the all-zero codeword is taken as the decoded word.

In the sequential algorithm and the bucket algorithm, the process of extending an entry of k elements to a codeword (finding the $n-k$ erasures) can be thought of as a decoding operation using erasures only decoding, or as an encoding operation. For a decoding operation the error magnitudes are found by solving a set of equations developed from the syndromes. The syndromes are found from the hard decision received word as

$$S_j = r(\alpha^j) = c(\alpha^j) + e(\alpha^j) = e(\alpha^j) = \sum_{k=0}^{n-1} e_k(\alpha^j)^k = \sum_{l=1}^{v} e_{i_l}\beta_l^j, j = 1, 2, \ldots, n-k, \tag{2.1}$$

where $r(x)$, $c(x)$, and $e(x)$ are the received, codeword, and error polynomials, respectively. By choosing the erasure locations the error locations, β_l, are determined and only the error magnitudes, e_{i_l}, must be found. Equation 2.1 can be rewritten as a matrix equation where the number of errors is known to equal the number of erasures, $\nu = n - k$,

$$\begin{bmatrix} \beta_1 & \beta_2 & \cdots & \beta_{n-k} \\ \beta_1^2 & \beta_2^2 & \cdots & \beta_{n-k}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \beta_1^{n-k} & \beta_2^{n-k} & \cdots & \beta_{n-k}^{n-k} \end{bmatrix} \begin{bmatrix} e_{i_1} \\ e_{i_2} \\ \vdots \\ e_{i_{n-k}} \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_{n-k} \end{bmatrix}. \tag{2.2}$$

This matrix is nonsingular since it is Vandermonde and can be solved directly, or by several different algorithms [5, 3].

Instead of decoding using erasure only decoding, the entries with k elements can be extended by encoding. The erasure locations determine the new data locations for encoding, and the data values are determined by the stack entries. To perform the encoding, a new generator matrix must be found which gives correct codewords for the new data locations. Let $\mathbf{G}$ be the systematic generator matrix used to encode the data for transmission. Another generator matrix, $\mathbf{G}'$, can be found which is a systematic generator for the data locations $\beta_1, \beta_2, \ldots, \beta_k$. Once the new generator matrix is found, the encoding process is simple matrix multiplication, $\mathbf{c} = \mathbf{u}\mathbf{G}'$ where $\mathbf{u}$ is the data sequence to be encoded.

The decoding process does not have the overhead of finding the new generator matrix, so for a small number of entries, the decoding process is more efficient than encoding. If several codewords are to be decoded, the encoding method is more efficient since the calculations required to perform the matrix multiplication overcome the complexity of finding $\mathbf{G}'$.

2.1.3 Soft-Decision Decoding Using an Extended Erasure Magnitude Matrix

Another method of generating a set of candidate codewords is by the use of the extended error magnitude matrix. The symbol locations are ordered by increasing reliability so that β_n is the most reliable symbol and $|m_{\beta_n,1}| > |m_{\beta_{n-1},1}| > \cdots > |m_{\beta_1,1}|$. The syndromes are defined as in Equation 2.1. The error magnitude matrix, Equation 2.2, is extended to allow for errors in the k non-erasure locations,

$$\begin{bmatrix} \beta_1 & \beta_2 & \cdots & \beta_{n-k} & \beta_{n-k+1} & \cdots & \beta_n \\ \beta_1^2 & \beta_2^2 & \cdots & \beta_{n-k}^2 & \beta_{n-k+1}^2 & \cdots & \beta_n^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \beta_1^{n-k} & \beta_2^{n-k} & \cdots & \beta_{n-k}^{n-k} & \beta_{n-k+1}^{n-k} & \cdots & \beta_n^{n-k} \end{bmatrix} \begin{bmatrix} e_{i_1} \\ e_{i_2} \\ \vdots \\ e_{i_n} \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_{n-k} \end{bmatrix}, \tag{2.3}$$

where e_{i_l} is the magnitude of the error at location β_l. By performing row operations, an equivalent equation is

$$\begin{bmatrix} 1 & 0 & \cdots & 0 & \beta'_{n-k+1} & \cdots & \beta'_n \\ 0 & 1 & \cdots & 0 & \beta^{(2)'}_{n-k+1} & \cdots & \beta^{(2)'}_n \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & \beta^{(n-k)'}_{n-k+1} & \cdots & \beta^{(n-k)'}_n \end{bmatrix} \begin{bmatrix} e_{i_1} \\ e_{i_2} \\ \vdots \\ e_{i_n} \end{bmatrix} = \begin{bmatrix} S'_1 \\ S'_2 \\ \vdots \\ S'_{n-k} \end{bmatrix}, \tag{2.4}$$

where $\beta_j^{(i)'}$, $i = 1, \ldots, n-k$, $j = n-k+1, \ldots, n$ and S'_j, $j = 1, 2, \ldots, n-k$ are functions of β_j, $j = n-k+1, \ldots, n$ and S_j, $j = 1, 2, \ldots, n-k$. The algorithm for finding the error magnitudes presented in [3] can be used to find the values for $\beta_j^{(i)'}$, $i = 1, \ldots, n-k$, $j = n-k+1, \ldots, n$ and S'_j, $j = 1, 2, \ldots, n-k$.

The first candidate codeword is obtained by using the hard-decision symbols for the data positions, $\beta_{n-k+1}, \ldots, \beta_n$. Since the hard-decision symbols are used in calculating the syndromes, there are no errors in the data positions, so $e_{i_l} = 0$ for $l = n-k+1, \ldots, n$. Thus, $e_{i_l} = S'_i$ for $i = 1, \ldots, n-k$.

Other candidate codewords are calculated by assuming different symbol values in the data positions, which is the same as assuming an error was made in the hard-decision for that location. Equation 2.4 can be rewritten to show the calculations needed to determine the different codewords. The new values for the erasures are found by

$$\begin{bmatrix} e_{i_1} \\ e_{i_2} \\ \vdots \\ e_{i_{n-k}} \end{bmatrix} = \begin{bmatrix} S_1' \\ S_2' \\ \vdots \\ S_{n-k}' \end{bmatrix} - e_{i_{n-k+1}} \begin{bmatrix} \beta_{n-k+1}' \\ \beta_{n-k+1}^{(2)'} \\ \vdots \\ \beta_{n-k+1}^{(n-k)'} \end{bmatrix} - e_{i_{n-k+2}} \begin{bmatrix} \beta_{n-k+2}' \\ \beta_{n-k+2}^{(2)'} \\ \vdots \\ \beta_{n-k+2}^{(n-k)'} \end{bmatrix} - \cdots - e_{i_n} \begin{bmatrix} \beta_n' \\ \beta_n^{(2)'} \\ \vdots \\ \beta_n^{(n-k)'} \end{bmatrix}, \tag{2.5}$$

where $e_{i_{n-k+1}}, e_{i_{n-k+2}}, \ldots, e_{i_n}$ are the new data symbols. For example, if the symbol value in location β_{n-k+1} is changed and the symbols in the other data locations are unchanged then $e_{i_{n-k+1}} \neq 0$ and $e_{i_{n-k+2}} = e_{i_{n-k+3}} = \cdots = e_{i_n} = 0$. The values for the erasures are now calculated as

$$\begin{bmatrix} e_{i_1} \\ e_{i_2} \\ \vdots \\ e_{i_{n-k}} \end{bmatrix} = -e_{i_{n-k+1}} \begin{bmatrix} \beta_{n-k+1}' \\ \beta_{n-k+1}^{(2)'} \\ \vdots \\ \beta_{n-k+1}^{(n-k)'} \end{bmatrix} + \begin{bmatrix} S_1' \\ S_2' \\ \vdots \\ S_{n-k}' \end{bmatrix}.$$

All possible codewords can be calculated in this manner.

To perform maximum-likelihood decoding, all possible codewords must be found and their corresponding codeword metrics calculated. The number of calculations needed for maximum-likelihood decoding is to large for most codes, but good results can be obtained by shortening the search. The most likely error events are those in which only a few errors occurred in the data locations, so all possible combinations of columns do not have to be considered. For example, considering only one of the extended columns at a time will correct all single errors outside of the $n-k$ erasures. Another way to shorten the search is to look at only the most likely symbol values for any location. Thus, codewords containing a symbol that has a low reliability at a data location are not considered.

2.1.4 Tree Algorithm

The tree algorithm is a method of using the extended error magnitude matrix to generate a set of candidate codewords. In the tree algorithm, unlike the sequential algorithm of Section 2.1.1, the longest data sequence will be extended as long as it has higher metric than any codeword even if it does not have the largest possible metric.

The tree algorithm is better thought of as moving through the branches of a tree instead of maintaining a stack. Again, the symbol locations are ordered by increasing reliability. Since β_n is the most reliable symbol location, it is the first node in the tree. Each of the first $k-1$ nodes have branches leaving the node. Each branch denotes the possible symbol value for the next entry in the data sequence. However, the k-th node has only one branch leaving the node, since at that point the codeword is completely determined by its data symbols.

The algorithm begins by initializing the first q branches which represent the possible symbol values for location β_n. Each node of the tree will have an associated metric which is the sum of the metrics of the symbols in the sequence. For the first node the metric is the metric of the most reliable symbol. The node with the largest metric branches to q nodes and new metrics are calculated for each of these nodes. After k branches the sequence is ready to decode. A new metric is calculated which includes the metrics of the parity symbols. If this metric is better than the metric at any other node, then this is the maximum-likelihood estimate of the codeword. If this metric is not the largest metric then the algorithm moves back to the previous node before the last branch. If any of the branches from that node have not been explored the algorithm follows the unexplored branch with the largest metric. If all of the branches have been explored, or a codeword metric is better than the remaining branches, the algorithm again moves back a node. The algorithm terminates when all paths from the initial node have been examined. A flow chart for this algorithm is shown in Figure 2.2.

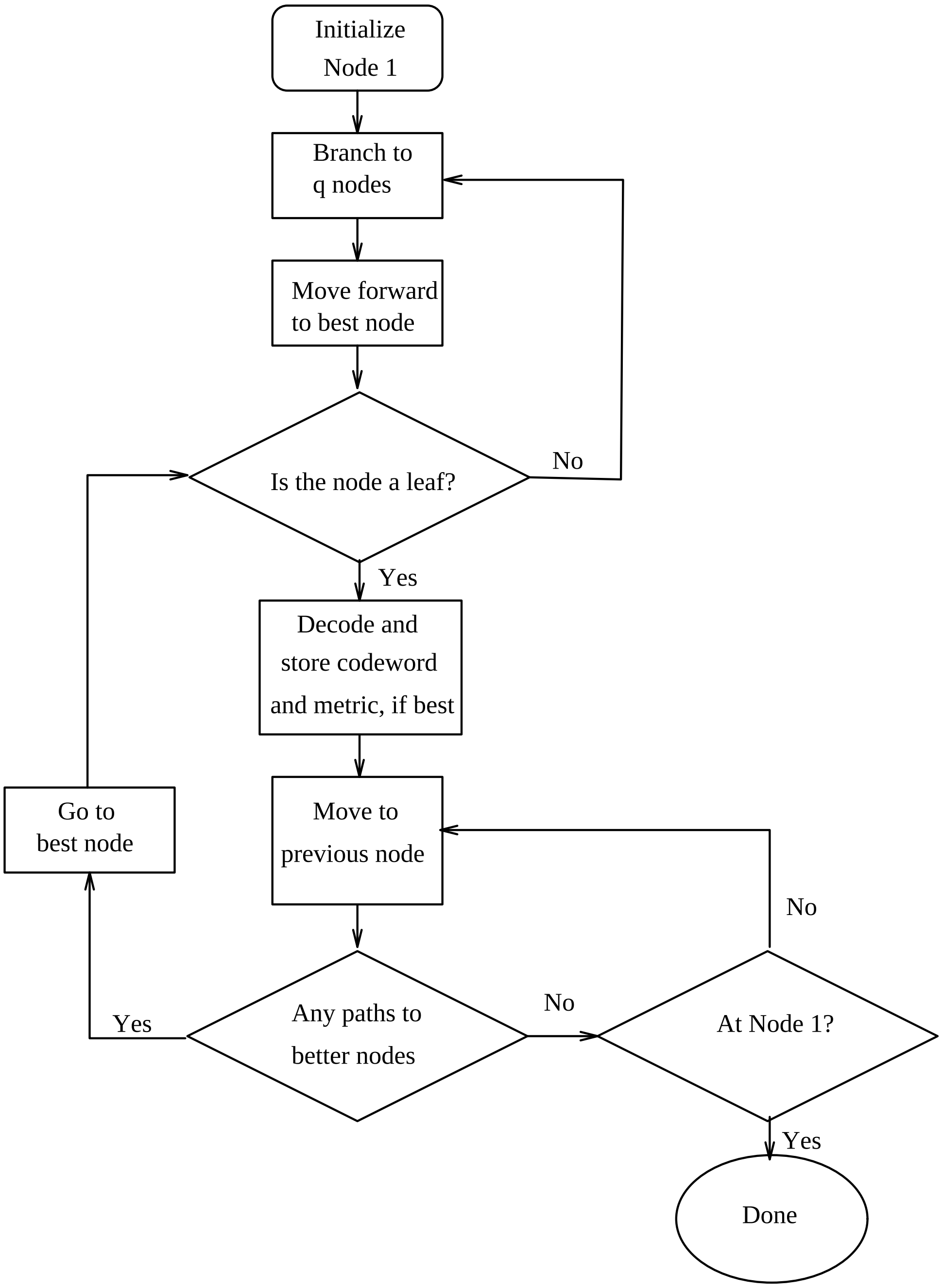

Figure 2.2 Tree Algorithm Flowchart

By extending the longest sequence with the largest metric, a stack is not required. Instead only the node on each branch that has been extended the farthest is stored. In addition, Equation 2.5 allows some of the calculations to be done as the data sequences are extended, which if stored prevents having to redo these calculations for every different sequence.

Example 4: To demonstrate the extended erasure decoding method using the tree algorithm, consider the following example of an (8,4) extended RS code. The codeword is transmitted over a channel using 8-ary FSK. The outputs of the matched filters are used to calculate the metrics for each symbol. The metrics for each symbol are given below.

Location 0		Location 1	
Symbol Value	Metric Value	Symbol Value	Metric Value
0	0.00	0	0.00
α^6	-3.69	α^3	-17.73
1	-4.47	α^4	-17.77
α	-4.58	α	-18.36
α^2	-4.72	α^2	-18.42
α^4	-4.96	α^5	-18.45
α^5	-5.11	1	-18.85
α^3	-5.33	α^6	-18.99
Location 2		**Location 3**	
Symbol Value	Metric Value	Symbol Value	Metric Value
0	0.00	α	0.00
1	-1.43	0	-1.37
α^6	-2.28	α^6	-1.43
α	-2.53	1	-2.23
α^3	-2.58	α^4	-3.22
α^2	-2.66	α^2	-3.46
α^5	-3.12	α^3	-3.51
α^4	-3.25	α^5	-3.54

Location 4		Location 5	
Symbol Value	Metric Value	Symbol Value	Metric Value
0	0.00	1	0.00
1	-2.26	0	-1.13
α^2	-3.30	α^5	-1.64
α^5	-3.45	α^4	-1.91
α	-4.58	α	-2.08
α^6	-4.97	α^6	-2.11
α^4	-5.10	α^3	-2.82
α^3	-5.26	α^2	-3.48
Location 6		Location 7	
Symbol Value	Metric Value	Symbol Value	Metric Value
α^2	0.00	0	0.00
0	-0.78	α^6	-1.25
1	-1.03	α^3	-1.57
α^5	-1.45	1	-2.32
α^3	-2.01	α^2	-3.01
α^6	-2.60	α^5	-3.24
α	-2.63	α^4	-3.87
α^4	-3.44	α	-4.02

Using the metrics, the ordering is obtained as $\beta_1 = 6, \beta_2 = 5, \beta_3 = 7, \beta_4 = 3, \beta_5 = 2, \beta_6 = 4, \beta_7 = 0$, and $\beta_8 = 1$. The hard decision received word is $r(x) = \alpha^2 x^6 + x^5 + \alpha x^3$, which is used to calculate the syndromes. From the syndromes and the location ordering, the extended erasure matrix is

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \alpha^6 & \alpha^5 & 0 & \alpha^3 & \alpha^2 & \alpha^4 & 1 & \alpha \\ \alpha^5 & \alpha^3 & 0 & \alpha^6 & \alpha^4 & \alpha & 1 & \alpha^2 \\ \alpha^4 & \alpha & 0 & \alpha^2 & \alpha^6 & \alpha^5 & 1 & \alpha^3 \end{bmatrix} \begin{bmatrix} e_6 & e_5 & e_7 & e_3 & e_2 & e_4 & e_0 & e_1 \end{bmatrix}^T = \begin{bmatrix} \alpha^5 \\ \alpha^3 \\ \alpha^3 \\ \alpha^2 \end{bmatrix}$$

where the first row of all 1's is for the zeroth syndrome, which is needed since this is an extended code [6]. Equivalently, the expression can be written as

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \alpha^6 & \alpha^6 & \alpha & \alpha^3 \\ 0 & 1 & 0 & 0 & \alpha^6 & \alpha^5 & \alpha^2 & \alpha^5 \\ 0 & 0 & 1 & 0 & \alpha & \alpha^2 & 1 & \alpha^4 \\ 0 & 0 & 0 & 1 & \alpha^3 & \alpha^5 & \alpha^4 & \alpha^3 \end{bmatrix} \begin{bmatrix} e_6 & e_5 & e_7 & e_3 & e_2 & e_4 & e_0 & e_1 \end{bmatrix}^T = \begin{bmatrix} \alpha^2 \\ 1 \\ 0 \\ \alpha \end{bmatrix}.$$

The tree is created by filling in the symbol values for each path. Each time only the node with the largest metric of the current branch is used to branch out to the next set of nodes. After the data sequence is four symbols long, the extended erasure matrix is used to decode. The tree at this point is given in Figure 2.3.

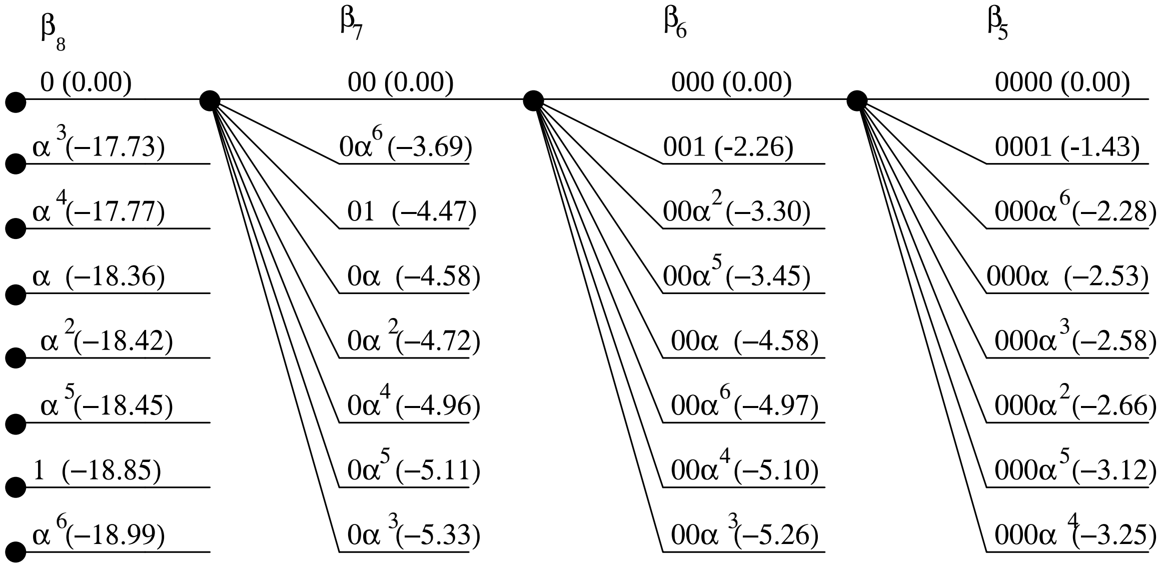

Figure 2.3 Tree Algorithm Example

The first set of decodings use the hard-decision values for data locations 1, 0, and 4, so $e_1 = e_0 = e_4 = 0$. The equation to find the erasure values is

$$\begin{bmatrix} e_6 \\ e_5 \\ e_7 \\ e_3 \end{bmatrix} = e_2 \begin{bmatrix} \alpha^6 \\ \alpha^6 \\ \alpha \\ \alpha^3 \end{bmatrix} + \begin{bmatrix} \alpha^2 \\ 1 \\ 0 \\ \alpha \end{bmatrix}.$$

All branches for this node are decoded simultaneously by evaluating the equation for all possible values of e_2. From the set of codewords produced by this equation, $c(x) = 0$ has the largest codeword metric at -3.28. The algorithm moves back one node and determines that the node with the sequence $001(-2.26)$ is the only node with a better metric than the codeword, so this node is branched out and another set of codewords is produced. For this set, the symbol value at location 4 is 1 so the erasure magnitude equation is

$$\begin{bmatrix} e_6 \\ e_5 \\ e_7 \\ e_3 \end{bmatrix} = (1) \begin{bmatrix} \alpha^6 \\ \alpha^5 \\ \alpha^2 \\ \alpha^5 \end{bmatrix} + e_2 \begin{bmatrix} \alpha^6 \\ \alpha^6 \\ \alpha \\ \alpha^3 \end{bmatrix} + \begin{bmatrix} \alpha^2 \\ 1 \\ 0 \\ \alpha \end{bmatrix} = e_2 \begin{bmatrix} \alpha^6 \\ \alpha^6 \\ \alpha \\ \alpha^3 \end{bmatrix} + \begin{bmatrix} 1 \\ \alpha^4 \\ \alpha^2 \\ \alpha^6 \end{bmatrix}.$$

The set of codewords produced by this equation have smaller metrics than the metric of the codeword already found, so decoding is complete. The estimate for the codeword is $c(x) = 0$.

2.2 Existing Algorithms

2.2.1 Errors and Erasures Decoding using Ratio Threshold Test

A simple method of using the reliability information from the receiver is to perform single pass errors and erasures decoding. The reliability information determines which symbols are unreliable and denotes them as erasures. If less than t non-erased symbols are in error where $t \leq \lfloor (d_{\min} - s)/2 \rfloor$ and s is the number of symbols erased, the received word will be decoded correctly. A simple method to determine which symbols should be erased is the ratio threshold test (RTT) [7, 8]. The RTT determines which symbols to erase on a symbol by symbol basis. If the channel outputs for a transmitted symbol are almost equal, then the detector cannot make a reliable decision as to which symbol was transmitted, so the symbol is erased. Let $\mathbf{y}_i = (y_{i,0}, y_{i,1}, \ldots, y_{i,q-1})$ be the channel output. The i-th received symbol is erased if

$$\frac{y_{i,j}}{y_{i,j_{max}}} > \gamma, j \neq j_{max}, \tag{2.6}$$

where $y_{i,j_{max}} = \max(y_{i,0}, y_{i,1}, \ldots, y_{i,q-1})$, and the threshold, γ is a fixed number in the interval $[0, 1]$. The number of erasures can be controlled by setting the threshold. A threshold close to one will allow very few erasures. As the threshold decreases, the number of erasures increases.

2.2.2 Generalized Minimum Distance Decoding

The RTT is an efficient method of using reliability information to improve errors only decoding performance. The generalized minimum distance decoding algorithm (GMD) introduced in [9] and improved on in [10], uses multiple passes of errors and erasures decoding to improve the performance from a single pass of an errors and erasures decoding.

The algorithm uses a reliability vector, $\boldsymbol{\alpha}_i = (\alpha_{i,0}, \alpha_{i,1}, \ldots, \alpha_{i,q-1})$ for each transmitted symbol, similar to that used in the soft-decision algorithms. The reliabilities are defined so that a larger value of $\alpha_{i,j}$ corresponds to a symbol, j, with a smaller likelihood of being the transmitted symbol. The components of $\boldsymbol{\alpha}_i$ are normalized so that the smallest component has value zero.

The algorithm makes use of an acceptance criterion which is based on a generalized distance measure. The extended distance of a codeword, $\mathbf{c}$, is defined as

$$d_e(\mathbf{c}, \mathbf{y}) = \sum_{i=1}^{n} \alpha(c_i, \mathbf{y}_i), \tag{2.7}$$

where $\alpha(a_j, \mathbf{y}_i)$ is a function which returns $\alpha_{i,j}$, the component of $\boldsymbol{\alpha}_i$ corresponding to the symbol value a_j. The algorithm is guaranteed to find a codeword with the smallest extended distance if it satisfies the acceptance criterion.

Each pass of the GMD algorithm performs an errors and erasures decoding. On the first pass no symbols are erased, so errors only decoding is performed. After each pass the resulting codeword is compared to an acceptance criterion. If the codeword is within the criterion, then decoding is complete. If not, then the number of erasures is increased by two and the process is repeated. The algorithm will find a codeword within the acceptance criterion if one exists.

There are two ways to implement the GMD algorithm. The algorithm can be implemented as described above, where it is possible that no codeword will be found that satisfies the acceptance criterion, and the decoder declares a decoding failure. Decoder failures are useful in systems which can request retransmissions in the event of a decoder failure. If a system cannot request retransmissions then it must make a decision as to which codeword was sent. In this situation the GMD algorithm can be used to generate a list of candidate codewords, and the decoder selects the codeword with the best metric. For purposes of comparison to the previous decoding algorithms, the latter implementation is used.

2.3 Results

The results of the new decoding algorithms are obtained by simulating a system using M-ary orthogonal signaling and noncoherent detection over a Rayleigh fading channel. The systems use a constellation size, M, equal to the code alphabet size. Extended RS codes are used, so the length of the code is the same as the code alphabet size, $n = q = M$.

A comparison of the existing and new decoding algorithms for an extended RS (32,12) code are shown in Figure 2.4. The performance of the decoding algorithms is compared based on the probability that a codeword is not decoded correctly. A decoder can either fail to decode or make a decoding error. Both are included in the probability of not decoding correctly. All of the algorithms discussed in Section 2.1 have a lower probability of not decoding correctly than errors only decoding. In addition, the soft decision decoding algorithms show an improvement over single pass errors and erasures decoding, which is to be expected since the decoder has more reliability information about the received symbols.

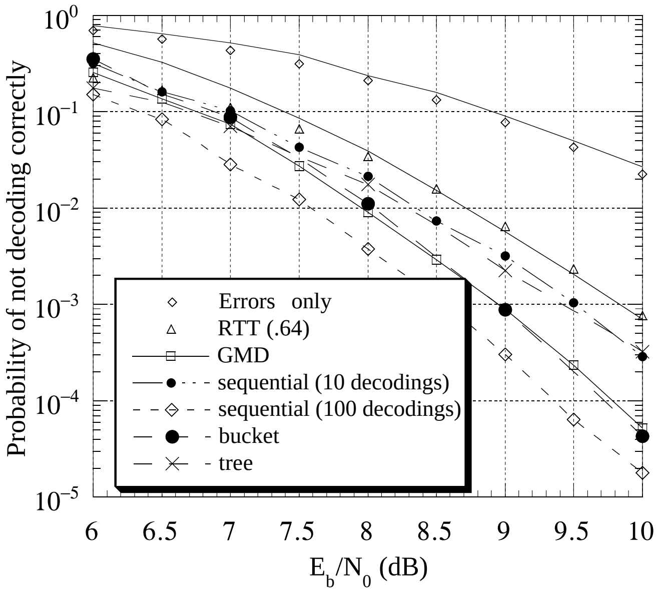

Figure 2.4 Comparison of Decoding Algorithms for RS (32,12)

The simulated GMD system does not perform true GMD decoding. For GMD decoding the reliability vector components are based on $\Pr(\mathbf{r}_i|\mathbf{y}_i)$, the probability that the vector $\mathbf{r}_i$ was received given that $\mathbf{y}_i$ was transmitted. Using this reliability vector, results that asymptotically approach maximum-likelihood decoding can be obtained. To calculate this reliability vector, channel information, such as noise and signal power is needed, which requires a more complex receiver. The reliability vector used for the new soft-decision decoding algorithms is based only on the outputs of the matched filters. Thus no extra information about the channel is needed, and a simpler receiver is used. In order to use the same receiver for both decoding methods, the reliability vector based on the filter outputs is used for the GMD decoding.

The sequential decoding algorithm performs better than the RTT even when the number of decodings is limited to ten. For the extended RS (32,12) code, the RTT can correct any combination of t errors and s erasures where $2t + s \leq 20$. The actual number of erasures is determined by the threshold. The threshold is set at 0.64 which provides the near-optimal results for the range of bit energy to noise ratios considered. With a higher threshold the decoder does not erase many symbols, so fewer errors are corrected and more decoder failures result. By lowering the threshold more erasures occur, but many erasures causes an increase in the number of decoder errors. The sequential decoding algorithm, when limited to ten decodings, considers up to ten different error patterns in the non-erased symbols, and chooses the best codeword from this set.

More improvement in the probability of not decoding correctly is obtained by increasing the maximum number of decodings for the sequential algorithm. By increasing the maximum number of decodings from ten to one hundred, approximately 1 dB of improvement is gained at a bit error rate of 10^{-3}. Neither of the simulations of the sequential decoding algorithm in this figure use a bias to decrease the average number of decodings.

The bucket algorithm used for Figure 2.4 also allowed a maximum of one hundred decodings and no bias. Ten buckets are used. The buckets used to store the sequences with the largest metrics (the top buckets) store a smaller range of metrics than the buckets that store the larger metrics (the bottom buckets). The use of smaller top buckets reduces the average number of decodings. Decoding is not completed until the bucket a codeword is in is emptied (all sequences in the bucket are examined). If buckets with a large range in metrics are used then many data sequences must be examined; however the use of smaller buckets reduces the number of data sequences in a given bucket. It is unlikely that a codeword will be in a bottom bucket, so allowing these buckets to be larger does not affect performance. The bucket algorithm does not perform as well as the sequential decoding algorithm with the same parameters, but it decodes faster since the stack does not have to be sorted.

The tree algorithm used for Figure 2.4 is limited to correct any combination of errors in the six least reliable data positions. These data positions have data values of the twelve most likely symbol values. In addition a bias of 0.75 is added to the symbols so that it is less likely that nodes will be extended once a codeword has been found.

Figure 2.5 shows the results of using the algorithms with a 16-ary FSK system using an extended RS (16,10) code. This code is a much weaker code than the (32,12) code. It can correct three errors if errors only decoding is used and six erasures with erasures only decoding. The improvement in going from errors only decoding to RTT is very slight since fewer additional errors can be corrected in a one pass errors and erasures algorithm. However, the soft-decision algorithms provide substantial improvement from the hard-decision algorithms.

The effects of adding a bias to the symbol metrics for the sequential decoding algorithm are shown in Figure 2.6. Again an extended RS (32,12) code is used with 32-ary FSK. The maximum number of decodings is limited to one hundred decodings and a bias of 0.0, 0.25, 0.50, or 0.75 is added to each metric. From Figure 2.6 it is seen that the effect of the bias is minimal to performance results. However, as can be seen in Figure 2.7, the average number of decodings performed for each received word is considerably reduced by using a bias.

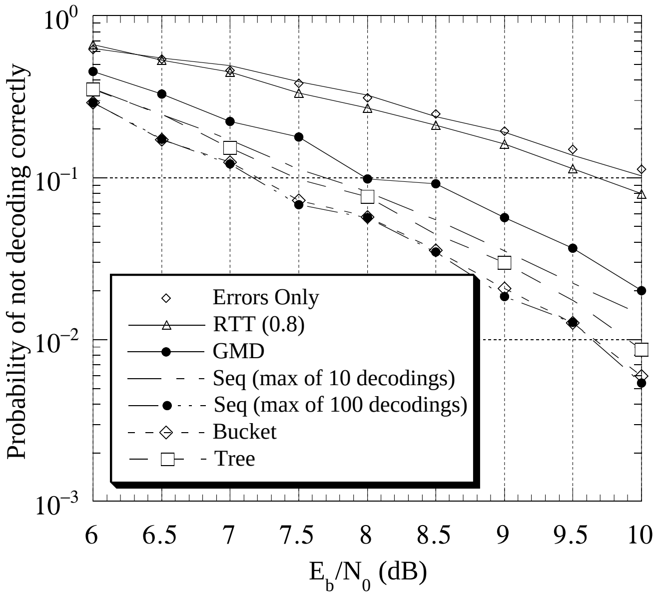

Figure 2.5 Comparison of Decoding Algorithms for RS (16,10)

The complexity in the tree algorithm is controlled by setting the number of error patterns outside of the erasure locations that will be considered. The system used for Figure 2.4 was allowed to correct any combination of errors that occur in the six least reliable data positions. Furthermore, the algorithm was only allowed to try the twelve most likely symbol values in these data positions. Figure 2.8 shows the resulting probability of error if the algorithm is only allowed to correct single or double errors outside of the erased positions. The curves marked with a S in the legend denotes that the algorithm can only correct single errors while a D denotes double error correction. In each trial the algorithm was allowed to consider twelve data positions (the maximum) or six data positions, and either 32 symbol values (the maximum) or sixteen symbol values. Better performance is obtained by considering

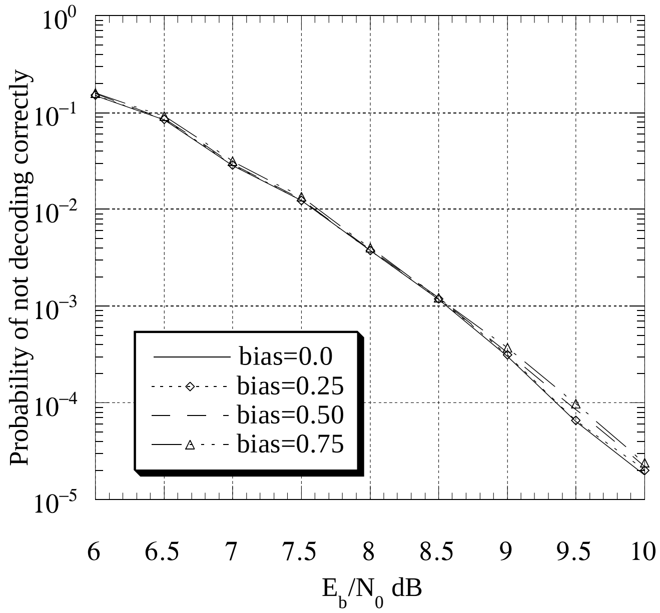

Figure 2.6 Effect of Bias on Sequential Decoding Algorithms

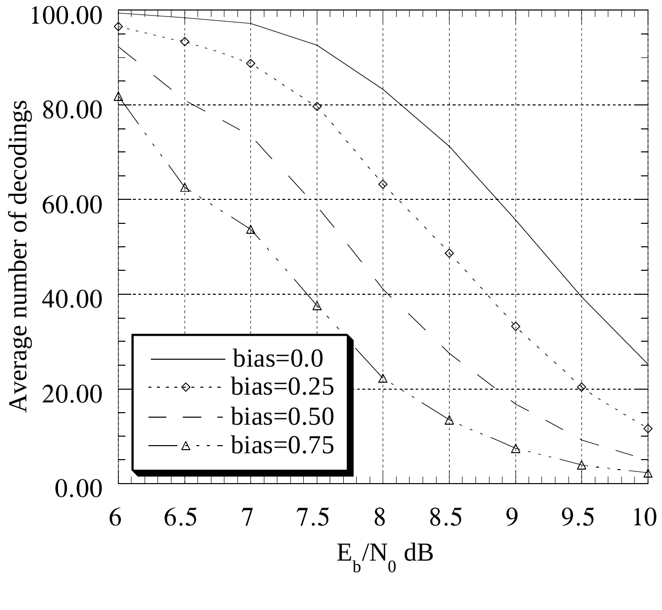

Figure 2.7 Effect of Bias on Reducing Number of Decodings

double error patterns when the number of error locations and symbol values used for each algorithm is the same. The algorithm is more sensitive to the number of symbol values considered rather than the number of error locations. For example the systems that only consider sixteen different symbol values, (S-6,16), (S-12,16), (D-6,16), and (D-12,16) all have worse performance than systems which consider all symbol error values, even if only single error patterns are considered.

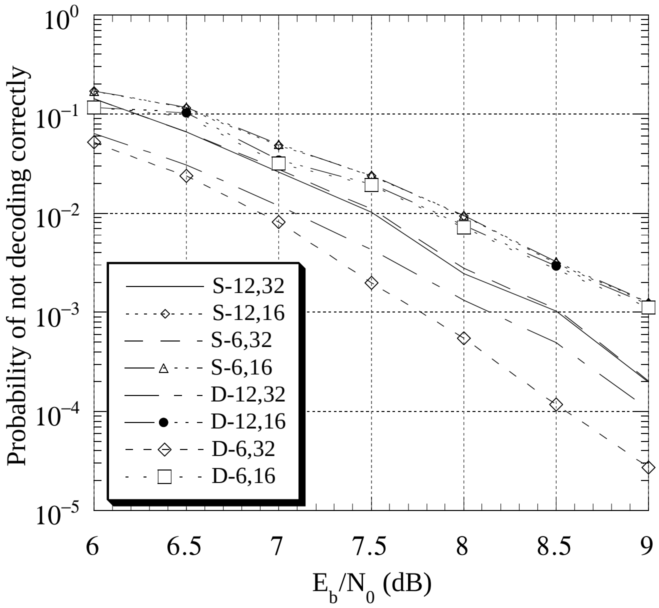

Figure 2.8 Reduced Search Tree Algorithm

2.4 Complexity

There are several complexity measures to consider. The algorithms can be compared on the basis of run time (or number of operations needed for a decoding), memory needed, and channel parameters required. The algorithms in this chapter do not rely on knowledge of the channel to determine the metrics. The memory needed for the two existing algorithms is similar to that of the tree algorithm since no stack is needed. The sequential algorithm and the bucket algorithm require the use of a stack, however the stack can be limited in size to a set maximum number of decodings allowed. The sequential algorithm also requires the stack be resorted after every decoding or extension which is not necessary in the other algorithms.

The main difference in the complexity of the algorithms results from the number of computations necessary for each of the algorithms. The computations can be counted as additions and multiplications of elements from the Galois field, GF(q). A field multiplication can be performed by simply taking the addition of the exponents of the symbols modulus $q-1$. A field addition is more complicated and requires the use of look up tables and either binary addition or field multiplication (if Zech's logarithms [1] are used). Since the Galois field operations are the main components of the decoding process, the complexity study calculates the numbers of field additions and multiplications needed to implement the above algorithms.

The sequential and bucket algorithms have the same complexity in terms of field operations. For either algorithm a data sequence must be extended to a codeword. The extension can either be accomplished by decoding the received word with $n-k$ erasures, or by encoding the k data symbols, as described in Section 2.1.2. If decoding is used the decoder must first calculate the $n-k$ syndromes and then find the $n-k$ erasure magnitudes for every codeword. If encoding is used the decoder must find a generator matrix for the new data locations. Once the generator matrix has been found, the set of codewords are produced by multiplying the data symbols by the generator matrix, $\mathbf{c} = \mathbf{mG}$. The totals of the field operations for each step of the algorithm are given in Table 2.1 for a (n, k) RS code over GF(q), where ν is the number of error and erasure locations, t is the number of errors, and s is the number of erasures.

The tree algorithm uses the fast algorithm [3] for finding the extended erasure magnitude matrix, but instead of finding one set of erasure magnitudes the algorithm must find the $(k+1)$ sets of magnitudes needed in Equation 2.4. Once the matrix is found then each codeword is found by Equation 2.5. The complexity can be reduced by looking at a limited number of symbols and only changing the data symbols in the most unreliable positions. If all error patterns in j locations are considered with m possible symbol values, then $(n-k)(m^j-1)$ field additions and field multiplications are needed to do the decoding. If the algorithm only considers double error patterns in the j locations, then $\binom{j}{2} m(n-k)(m+1)$ field multiplications and additions are required. The complexity can be further reduced by considering only single errors in the j locations. To correct all single errors, $mj(n-k)$ field multiplications and additions are required.

Table 2.1 Complexity of Algorithm Components

operation	field additions	field multiplications
syndromes	$(n-k)(n-1)$	$(n-k)(n-1)$
error magnitudes	$3\nu(\nu-1)/2$	ν^2
new generator matrix	$k(k-1)(2(n+1)-k)/2$	$k(k-1)(2(n+1)-k)/2$ $+k(n-k+1)$
encode data sequence	nk	nk
extended error magnitude matrix	$3(k+1)(n-k)(n-k-1)/2$	$(k+1)(n-k)^2$
find $\Lambda(x)$ (errors only)	$2t(t+1)$	$3t(t+1)$
find roots of $\Lambda(x)$	qt	qt
find $\Gamma(x)$	$\sum_{j=1}^{s} \binom{s}{j}$	$\sum_{j=1}^{s} \binom{s}{j} (j-1)$
find $\Lambda(x)$ with s erasures	$d_{\min}^2 - d_{\min} - s^2 - 3$	$d_{\min}^2 - s^2/2 - s - 1$
find all $\Lambda(x)$ using new algorithm	$15t^2/2$	$15t^2/2$

The number of calculations needed for errors only decoding and single pass errors and erasures decoding are given in Table 2.1, where $\Lambda(x)$ is the error locator polynomial and $\Gamma(x)$ is the erasure locator polynomial. The error locator polynomial is found by using the Berlekamp-Massey algorithm [1]. Once $\Lambda(x)$ is calculated the roots, which are the inverses of the error locations, are found by using Chien's algorithm [4].

In GMD decoding using the Berlekamp-Massey algorithm [11] the errors and erasures decoding must be performed up to $\lfloor (n-k)/2 \rfloor$ times. Not all of the calculations have to be repeated for each step. The syndromes have to be calculated only once per received word, and the erasure locator polynomial can be updated with every new set of erasures.

Recent research [12, 13, 14, 15] has produced new decoding algorithms that decrease the number of field calculations for GMD decoding of RS codes. These algorithms use an iterative approach to generated the error locator polynomials for each combination of erasures. One such algorithm [12] needs $15t^2/2$ field multiplications and additions to calculate all $\lfloor (n-k)/2 \rfloor$ error locator polynomials. The other algorithms require similar calculations. Once the error locator polynomials are found, the error locations and error magnitudes are calculated. The new algorithms greatly reduce the number of field calculations needed for GMD decoding, and only require a small amount of additional memory.

Table 2.1 can be used to calculate the numbers of computations required by the algorithms that are compared in Figure 2.4. For the comparison the maximum number of decodings is used for the GMD, sequential, and tree algorithms. The results are given in Table 2.2. The number of calculations needed for errors only decoding is included for reference. The number of calculations for RTT is determined by the number of erasures for a given received word. The worst case situation is for eighteen erasures and one error, and this number of calculations is listed in the table. The Tree-A entry considers all possible combinations of error patterns in six locations. This algorithm was used for the curve in Figure 2.4. The Tree-D and Tree-S entries consider double and single error patterns in six locations. The performance results from these algorithms are shown in Figure 2.8.

Table 2.2 Complexity of Soft-Decision Algorithms

Algorithm	Additions	Multiplication
Errors Only	1.295×10^3	1.390×10^3
RTT	1.969×10^3	2.002×10^3
GMD	2.109×10^6	1.888×10^7
New GMD	9.830×10^3	8.785×10^3
Sequential (10)	7.140×10^3	7.392×10^3
Sequential (100)	4.170×10^4	4.195×10^4
Tree-A(12,6)	5.973×10^7	5.973×10^7
Tree-D(32,6)	3.168×10^5	3.168×10^5
Tree-S(32,6)	3.840×10^3	3.840×10^3

The sequential algorithm with 100 decodings provides the best performance at the cost of more field computations than GMD decoding or the tree algorithm correcting only single errors. The tree algorithm correcting only single errors provides slightly better performance than GMD decoding with fewer field computations. In order to correct all double error patterns outside of the erasures, the tree algorithm requires a factor of 100 more field computations, but as can be seen in figure reffig:data5 better performance is achieved.

CHAPTER 3

CYCLIC AND SHORTENED CYCLIC CODES

3.1 Nonbinary BCH Codes

BCH codes are a special class of cyclic codes. A q-ary BCH code of length n is found in the same field that contains the q-ary cyclic codes of length n, namely GF(q^m) where m is minimal such that $n|(q^m - 1)$.

If the generator polynomial has roots $R = \{\gamma^{r_1}, \gamma^{r_2}, \ldots, \gamma^{r_m}\}$ where $\gamma = \alpha^{q^m-1/n}$ and $\alpha \in$GF(q^m), α primitive, the design distance, δ, of the BCH code is one more than the number of consecutive powers of γ in R. It can be shown [1] that $d_{min} \geq \delta$. Since the minimum distance of a BCH code may be larger than the design distance of the code, and since there are cyclic codes which may not be BCH codes, cyclic codes can exist with larger minimum distances than the design distance of a BCH code with the same parameters. While the design distances of BCH codes are known from the roots of the generator polynomial, the minimum distance of a cyclic code must be found by a search over all codewords of the code.

Example 1: A length 85, 4-ary code is defined over the field GF(4^4). An $(85, 81)_4$ code is found that has roots $\{\gamma^7, \gamma^{28}, \gamma^{27}, \gamma^{23}\}$ where $\gamma = \alpha^3$. The generator polynomial is $g(x) = m_{21}(x) = x^4 + \beta x^3 + \beta^2 x^2 + \beta^2 x + 1$, where $m_{21}(x)$ indicates the minimal polynomial with root $\gamma^7 = \alpha^{21}$, $\alpha \in$GF(4^4), α primitive, and $\beta = \alpha^{85} \in$GF(4). This code is a BCH code with a design distance $\delta = 3$. Searching over all possible $(85, 81)_4$ codes, the best codes also have distance $d_{min} = 3$. However, the best BCH $(85, 77)_4$ code has design distance $\delta = 4$, but an $(85, 77)_4$ code can be found with roots $\{\gamma, \gamma^4, \gamma^{16}, \gamma^{64}, \gamma^{42}, \gamma^{93}, \gamma^{77}, \gamma^{53}\}$ that has distance $d_{min} = 5$.

3.2 Shortened Cyclic Codes

A code can be shortened by deleting l message coordinates, which shortens an (n, k) code to an $(n - l, k - l)$ code to produce a code of any desired length. The generator matrix, $\mathbf{G}'$, of the shortened code is obtained by deleting rows and columns of the original matrix, $\mathbf{G}$. The minimum distance of the shortened code is greater than or equal to the minimum distance of the original code [1]. Shortened cyclic codes can be decoded using the same algorithms as cyclic codes.

A q-ary primitive polynomial generates a sequence of period $q^m - 1$, where $\mathrm{GF}(q^m)$ is the field the polynomial is defined over. The sequence contains all possible q-ary vectors of length m. Thus, $(1, 0, 0, \ldots, 0)$ is a vector in the sequence as is $(\gamma^i, 0, 0, \ldots, 0)$, where $\gamma^i \in \mathrm{GF}(q)$, $\gamma^i \neq 1$. Even though the sequence has a period of $q^m - 1$, it repeats to a multiple of the sequence at $(q^m - 1)/(q - 1)$ time shifts. Therefore, the sequence can be thought of as having an effective period, p_e, of $(q^m - 1)/(q - 1)$. The contents of the registers at each time shift is a column of the parity check matrix, $\mathbf{H}$. A code using this polynomial as a generator has a minimum distance of at most two, since the first and the $[(q^m - 1)/(q - 1)] + 1$ columns of the parity check matrix are linearly dependent.

A q-ary cyclic code of length n is constructed by generating a field of $\mathrm{GF}(q^m)$ where m is the minimal m such that $n|(q^m - 1)$. To find a generator polynomial, roots are needed with order n with respect to $\mathrm{GF}(q^m)$. The elements of the field $\mathrm{GF}(q^m)$ have an order which divides $q^m - 1$. If $q^m - 1 = p_1^{i^1} p_2^{i^2} \ldots p_r^{i^r}$ where p_i is a prime, then q-ary cyclic codes are found over $\mathrm{GF}(q^m)$ which have length n where n is a product of various powers of the primes, $p_1, p_2, \ldots, p_r$. Since $q - 1$ is a divisor of $q^m - 1$, there are elements in $\mathrm{GF}(q^m)$ that have order $q - 1$ and multiples of $q - 1$. If a generator polynomial is used that has roots with order s where $(q - 1)|s$, the sequence generated by this polynomial repeats to a multiple of the original sequence after $s/(q - 1)$ time shifts for an effective period of $p_e = s/(q - 1)$. Thus, good cyclic codes cannot be generated with length n.

To avoid the problem of generator polynomials which generate sequences that have an effective period less than the length of the code, a longer code is generated and shortened to the desired length. Assume a q-ary code of length n is desired where

$(q-1)$ divides n, which is denoted as $(q-1)|n$. If a code of length $n(q-1)$ can be found in the same field, GF(q^m) as the original length n code, this code is used and shortened to length n. A code of length $n(q-1)$ has a generator polynomial with an effective period of n, and an actual period of $n(q-1)$. The code is shortened to length n by deleting the last $n(q-2)$ message coordinates. By deleting the corresponding columns in the parity check matrix, the linearly dependent columns are removed and the shortened code has better distance properties than the original code.

In general the minimum distance of a shortened cyclic code is found by searching over all codewords of the code. However, a length $n = (q^m - 1)/(q-1)$ code with $k = n - m$ over GF(q^m) exists with $d_{min} = 3$, and is a nonbinary Hamming code, as shown in Theorem 1.

Theorem 1 *A shortened cyclic q-ary (nonbinary, q a prime power) single-error correcting Hamming code with length $n = (q^m - 1)/(q-1)$ with $n - k = m$ parity symbols, is obtained by shortening a length $n(q-1)$ cyclic code generated from a degree m monic primitive polynomial $g(x) \in GF(q)[x]$. The weight enumerator of the dual of this code is given as $A^{\perp}(x) = B(x) = 1 + (q^m - 1)x^{q^{m-1}}$.*

Proof: Form a parity check matrix $\mathbf{H}$ for a q-ary code of length $(q-1)n = q^m - 1$ cyclic code with $n - k = m$, using the degree m monic primitive polynomial $g(x)$. The columns of $\mathbf{H}$ consist of all the distinct q-ary m-tuples except the all-zero one. Thus, each row of $\mathbf{H}$ has $q^m(q-1)/q = (q-1)q^{m-1}$ non-zero elements since all q elements are equally distributed (each row is an m-sequence). For each column of $\mathbf{H}$ (each q-ary m-tuple) there are $q-2$ other distinct nonzero m-tuples that are multiples of that m-tuple and as such are pairwise linearly dependent. If $\mathbf{H}$ is obtained from the contents of a linear feed-back shift register connected according to $g(x)$, the linear dependent columns can be eliminated by removing the $(q^m - 1)(q-2)/(q-1)$ rightmost columns, thereby shortening the code to a length $n = (q^m - 1)/(q-1)$ code. Since the columns of the shortened $\mathbf{H}$, $\mathbf{H}'$, are distinct and not multiples of any other column, the code corresponding to $\mathbf{H}'$ has minimum distance 3. Thus, $\mathbf{H}'$ is a parity check matrix for a single-error correcting code. Now each row of $\mathbf{H}'$, or the generator matrix for the dual code $\mathbf{G}^{\perp}$, has q^{m-1} non-zero elements and the weight of all of the basis vectors for the dual code, $\mathbf{C}^{\perp}$, is q^{m-1}. Any $\mathbf{c}_i^{\perp} \in \mathbf{C}^{\perp}$ can be written $\mathbf{c}_i^{\perp} = \sum_{j=0}^{m-1} \mathbf{a}_j \mathbf{g}_j^{\perp}$ which is a $1/(q-1)$ segment of an m-sequence, but $\mathbf{c}_i^{\perp}$ and $(m-1)$ $\mathbf{g}_i^{\perp}$'s can be used as a basis for $\mathbf{C}^{\perp}$ or rows of $\mathbf{G}^{\perp}$. Since each row of $\mathbf{G}^{\perp}$ must be a $1/(q-1)$ segment of an m-sequence, $\mathbf{c}_i^{\perp}$ must have weight q^{m-1}. Thus, all non-zero codewords have weight q^{m-1}, or the weight enumerator for the (n, m) dual code is $A^{\perp}(x) = B(x) = 1 + (q^m - 1)x^{q^{m-1}}$.

Example 2: A length 21, 4-ary code is defined over the field GF(4^3). If roots of order 21 are used, the generator polynomial has an effective period of seven. For example, if the roots $\{\alpha^3, \alpha^{12}, \alpha^{48}\}$ are used, the generator polynomial is $g(x) = x^3 + \gamma^2 x + 1$ where $\gamma = \alpha^{21} \in$GF(4). The parity check matrix for this code is:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & \gamma^2 & 1 & \gamma & 0 & 0 & \gamma & 0 & 1 & \gamma & \gamma^2 & 0 & 0 & \gamma^2 & 0 & \gamma & \gamma^2 \\ 0 & 1 & 0 & \gamma^2 & 1 & \gamma & 0 & 0 & \gamma & 0 & 1 & \gamma & \gamma^2 & 0 & 0 & \gamma^2 & 0 & \gamma & \gamma^2 & 1 & 0 \\ 0 & 0 & 1 & 0 & \gamma^2 & 1 & \gamma & 0 & 0 & \gamma & 0 & 1 & \gamma & \gamma^2 & 0 & 0 & \gamma^2 & 0 & \gamma & \gamma^2 & 1 \end{bmatrix}.$$

The first and eighth columns are linearly dependent, so the minimum distance of the code is two. However, if a code of length 63 with roots of order 63 is shortened to length 21, the shortened code has a minimum distance of three, by Theorem 1.

If a code of length $n(q-1)$ cannot be found in the same field as the original length n code, a length n' code, $n' > n$, is used that has $p_e \geq n$. The distance properties of the shortened code are found by a search procedure.

3.3 Comparison of BCH, Cyclic, and Shortened Cyclic Codes

Tables 3.1-3.3 show examples of BCH, cyclic, and shortened cyclic codes. The tables also show either the design distance for the BCH codes, or the minimum distance for cyclic and shortened cyclic codes. Since BCH codes are cyclic codes and the minimum distance of a BCH code may be larger than the design distance, the minimum distance of a cyclic code may be larger than the design distance for the BCH code. For the codes in these tables that have length n such that $n(q-1)|(q^m-1)$, shortened codes are created by shortening a length $n(q-1)$ code to a length n code. If $n(q-1)$ does not divide (q^m-1), the shortened codes are obtained by shortening a code of length n', $n' > n$, that is defined over the same field as the length n code, GF(q^m). These shortened codes have minimum distances that are at least as large as the comparable cyclic codes.

The gains, in terms of improvement of minimum distance, in going from BCH or cyclic codes to shortened cyclic codes, are for codes whose length is divisible by a factor of $(q-1)$. If the length of a BCH or cyclic code is divisible by a factor of $(q-1)$, the effective period of the code is less than the code length and poor distance properties exist. However, if a longer code is used instead and shortened to the desired length, the shortened code has better distance properties, as is seen in the tables.

3.4 Decoding Algorithms for Cyclic Codes

3.4.1 Hard-Decision Decoding of Cyclic Codes

In general cyclic codes do not have efficient decoding algorithms. The BCH codes and Reed-Solomon codes are guaranteed to have $2t$ linearly independent equations if certain syndromes are used. By using the correct syndromes, up to t hard-decision errors can be corrected using efficient algorithms such as the Berlekamp-Massey algorithm. The maximum numbers of errors that can be corrected is t,

$$t = \left\lfloor \frac{(d-1)}{2} \right\rfloor . \tag{3.1}$$

For RS codes, $d = d_{min} = n - k + 1$, and for BCH codes d is the design distance of the code, δ, which is one more than the number of consecutive roots of the generator polynomial.

Table 3.1 4-ary BCH, Cyclic, and Shortened Cyclic Codes

Field	Code	BCH δ	Cyclic d_{min}	Shortened Cyclic d_{min}
GF(4^3)	(63,59)	3	3	not possible
	(63,54)	5	5	not possible
	(21,18)	2	2	3
	(21,15)	3	3	5
	(21,12)	5	5	7
	(21,9)	7	7	9
	(9,6)	2	2	3
	(9,4)	3	3	4
	(9,3)	3	3	6
	(9,2)	5	5	7
	(7,4)	3	3	3
GF(4^4)	(85,81)	3	3	3
	(85,79)	3	4	4
	(85,77)	4	5	5
	(85,73)	6	6	6
	(85,71)	5	5	6
	(51,47)	2	2	3
	(51,43)	3	3	5
	(51,39)	5	5	6
	(17,13)	3	4	4
	(17,9)	7	7	7
	(17,8)	7	8	8
	(17,5)	9	9	9
	(17,4)	9	12	12
	(15,11)	3	3	4
	(15,7)	7	7	7
GF(4^5)	(341,336)	3	3	3
	(341,331)	4	5	5
	(93,88)	2	2	3
	(93,87)	3	3	3
	(93,85)	3	3	4
	(93,83)	3	3	5
	(33,28)	2	2	3
	(33,23)	3	3	6
	(31,26)	3	3	3
	(31,21)	5	5	6
	(31,6)	15	15	15
	(11,6)	4	5	5
GF(4^6)	(195,189)	2	2	3
	(195,183)	3	3	4
	(195,6)	131	132	140

Table 3.2 8-ary BCH, Cyclic, and Shortened Cyclic Codes

Field	Code	BCH δ	Cyclic d_{min}	Shortened Cyclic d_{min}
GF(8^3)	(73,70)	3	3	3
	(73,67)	4	4	4
	(73,66)	4	5	5
GF(8^4)	(195,191)	3	3	3
	(195,187)	4	4	4
	(45,41)	3	3	3
	(45,37)	4	4	4
	(35,31)	2	2	3
	(35,27)	3	3	5
GF(8^7)	(127,120)	3	3	3
	(127,8)	63	63	92
	(127,7)	63	64	92

Table 3.3 16-ary BCH, Cyclic, and Shortened Cyclic Codes

Field	Code	BCH δ	Cyclic d_{min}	Shortened Cyclic d_{min}
GF(16^3)	(91,88)	3	3	3
	(91,85)	4	5	5
	(63,60)	2	2	3
	(63,57)	3	3	5
	(39,36)	2	2	3
	(39,33)	3	3	5

3.4.2 Soft-Decision Decoding of Cyclic Codes

In general cyclic codes are not maximum distance separable codes. In the algorithms of Chapter 2, a property of MDS codes is used to decode RS codes. This property states that any k locations in the codeword can be used as data locations. For a general cyclic code, this property is not true, so the algorithms of Section 2.1.1 and Section 2.1.2 cannot be used to decode these codes. However, with some modifications, the extended error magnitude matrix can be used to decode general cyclic codes similar to decoding binary BCH codes [16].

Let β_i be the location of the i-th most reliable symbol location. Thus β_1 is the least reliable location and β_n is the most reliable location. The syndromes used in decoding are defined as usual,

$$S_j = r(\alpha^j) = c(\alpha^j) + e(\alpha^j) \tag{3.2}$$

For RS codes, the roots of the codes are the first $n-k$ roots, so $c(\alpha^j) = 0$ for $j = 1, 2, \ldots, n-k$. For the generalized cyclic codes, the set of conjugacy classes which contains all the roots of the generator polynomial is given as $P = \{p_1, p_2, \ldots, p_{n-k}\}$, so $c(\alpha^j) = 0$ only when $j \in P$. Thus,

$$S_j = e(\alpha^j) = \sum_{i=0}^{n-1} e_l(\alpha^j)^i = \sum_{l=1}^{\nu} e_{i_l}\beta_l^j \text{ for } j \in P. \tag{3.3}$$

In matrix notation an augmented equation to obtain the error magnitudes is given as

$$\begin{bmatrix} \beta_1^{p_1} & \beta_2^{p_1} & \cdots & \beta_{n-k}^{p_1} & \beta_{n-k+1}^{p_1} & \cdots & \beta_n^{p_1} \\ \beta_1^{p_2} & \beta_2^{p_2} & \cdots & \beta_{n-k}^{p_2} & \beta_{n-k+1}^{p_2} & \cdots & \beta_n^{p_2} \\ \cdots & \cdots & \ddots & \cdots & \cdots & \ddots & \cdots \\ \beta_1^{p_{n-k}} & \beta_2^{p_{n-k}} & \cdots & \beta_{n-k}^{p_{n-k}} & \beta_{n-k+1}^{p_{n-k}} & \cdots & \beta_n^{p_{n-k}} \end{bmatrix} \begin{bmatrix} e_{i_1} \\ e_{i_2} \\ \vdots \\ e_{i_n} \end{bmatrix} = \begin{bmatrix} S_{p_1} \\ S_{p_2} \\ \vdots \\ S_{p_{n-k}} \end{bmatrix}, \tag{3.4}$$

or in a shortened notation

$$[\beta|\mathbf{P}]\,\mathbf{e} = \mathbf{S}. \tag{3.5}$$

where $\boldsymbol{\beta}$ consists of the $n-k$ leftmost columns and $\mathbf{P}$ the remaining columns.

The powers of β_i in the matrix $\boldsymbol{\beta}$ are no longer consecutive powers, so the matrix is not Vandermonde, and $\mid \boldsymbol{\beta} \mid$ may equal 0 indicating a set of linearly dependent equations. A set of linearly independent equations is found by interchanging the rightmost columns of $\boldsymbol{\beta}$ with the leftmost columns of $\mathbf{P}$ to form a matrix $\boldsymbol{\beta}'$ such that $\mid \beta' \mid \neq 0$. With the new matrices Equation 3.5 becomes

$$[\boldsymbol{\beta}'|\mathbf{P}'] \, \mathbf{e} = \mathbf{S}. \tag{3.6}$$

Equation 3.6 can be solved as in Section 2.1.4, or solved just for single errors outside of the erased positions. Since columns of $\boldsymbol{\beta}$ are interchanged with columns of $\mathbf{P}$, not all of the erased positions are the most likely positions to contain errors, but the effect on performance is small. This method of decoding decodes up to the true minimum distance of the code.

CHAPTER 4

COMPARISON OF CODED M-ARY ORTHOGONAL COMMUNICATION SYSTEMS

Normally for error control on a communication system using M-ary orthogonal signaling, a Reed-Solomon code of length n, which is equal to the size of the signal constellation is used. A larger signal constellation requires a more complex receiver system. For example with M-ary orthogonal signaling and noncoherent detection, $n = M$ envelope detectors or $2M$ squaring devices are needed to implement the M-ary receiver. In order to reduce the complexity of the receiver, it is desired to reduce the signal constellation size. However, a larger constellation size for an uncoded system results in a lower probability of bit error than if a smaller signal set is used. Therefore, to compensate for this loss of performance error correction codes are utilized. A system using a signal constellation of size M' where $M' < M$ can be found that uses multiple channel symbols to represent one code symbol. Since a smaller signal constellation is used, a gain in bandwidth is obtained. To keep the bandwidth requirements the same for a fair comparison between the two systems, the code rate of the system using the M' constellation size is decreased which increases the amount of redundancy used. The added redundancy provides better error control performance for the system.

4.1 System Description

Consider a system using M-ary orthogonal signaling where noncoherent demodulation is used. The channel is a non-selective Raleigh fading channel. From [17, 18] the probability of receiving a channel symbol in error is

$$p_{ce} = M^{-1} \sum_{j=2}^{M} \binom{M}{j} (-1)^j \left[1 + (1 - j^{-1}) E_s/N_0\right]^{-1}, \tag{4.1}$$

where E_s/N_0 is the symbol energy to noise ratio. For a fair comparison between systems of different channel symbol alphabets, the symbol energy is converted to bit energy by $E_s = E_b \log_2 M$. Error correcting codes introduce redundancy, which increases the channel symbol transmission rate and reduces the received energy per symbol by the code rate, R, where $R = k/n$.

For systems where the signal constellation size, M, is the same as the code alphabet size, q, one code symbol is sent as a channel symbol. Therefore, the probability of code symbol error is the same as the probability of channel symbol error. If $q > M$and $\log_2 q/\log_2 M$ is an integer, then a code symbol must be transmitted as a series of $\log_2 q/\log_2 M$ channel symbols. Any channel symbol error in the sequence will produce a code symbol error, so the probability of a code symbol error is, from [1],

$$p_e = 1 - (1 - p_{ce})^{\log_2 q/\log_2 M}, \tag{4.2}$$

asumming the channel symbol errors are independent and an integer number of channel symbols are in a code symbol. If there are a noninteger number of channel symbols in a code symbol, then a code symbol will contain fractions of channel symbols, and the code symbol error will depend on the arrangement of the channel symbols within the code symbols.

4.2 Equivalent System Comparison

Let System 1 be a traditional system using M_1-ary signaling and a (n_1, k_1) extended Reed-Solomon code. For this system the code symbol alphabet size of q_1 is the same as the channel symbol alphabet size of M_1, so each code symbol is sent as one channel symbol. The length of the code, n_1, for an extended RS code is the same as the code alphabet size, q_1, and $R_1 = k_1/n_1$.

Let System 2 use M_2-ary signaling where $M_2 < M_1$. The code symbol alphabet is kept the same, so the code is now a RS (n_1, k_2) code. Since a smaller channel symbol set is being used, less bandwidth is needed, thus the code rate of the second code, R_2, can be decreased. The bandwidth requirement for an M_2-ary system is less by a factor of M_2/M_1 due to the smaller signaling constellation, so the rate of the code for System 2 can be reduced by this factor. However, $\log_2 M_1/\log_2 M_2$ channel symbols must be sent for each code symbol, so the rate must be increased by this factor. The resulting rate is $R_2 = k_2/n_1$ where

$$k_2 = k_1 \left(\frac{M_2}{M_1}\right) \left(\frac{\log_2 M_1}{\log_2 M_2}\right). \tag{4.3}$$

Other systems can be found using different signal constellations as long as the resulting rate produces a valid code. For some choices of M_1, k_1, and M_2 a fractional k_2 results, which implies that no RS code of that size exists that can be fairly compared to the original system.

A third system, System 3, is also considered if $\log_2 M_1/\log_2 M_2$ is an integer, or if n_3 and k_3 are integers where

$$\begin{aligned} n_3 &= n_1 \tfrac{\log_2 M_1}{\log_2 M_2}, \text{ and} \\ k_3 &= k_1 \tfrac{M_2}{M_1} \left(\tfrac{\log_2 M_1}{\log_2 M_2}\right)^2. \end{aligned} \tag{4.4}$$

This system can be developed using a code alphabet of size M_2 instead of M_1. System 3 uses the same signal constellation size as System 2, so the bandwidth requirements are the same, and thus, the code rate is kept the same. Since a code symbol and a channel symbol are the same for this system, $\log_2 M_1/\log_2 M_2$ code symbols are transmitted using this system for every one code symbol transmitted with System 2. Therefore, the code length can be increased by this factor, and $n_3 = n_1(\log_2 M_1/\log_2 M_2)$. To maintain the same code rate, the data length, k_3, is also increased by this factor. Since the code length is no longer equal to the alphabet size, a Reed-Solomon code cannot be used. Another code, such as a nonbinary cyclic code, with the desired length and rate is needed.

4.3 Examples of Equivalent Systems

Consider a system using 64-ary orthogonal signaling and a (64,60) extended Reed-Solomon code which corrects double errors and is defined over GF(64). If instead 32-ary orthogonal signaling is used, a (64,36) extended RS code is used. This code can correct fourteen code symbol errors, where the code symbols are still defined over GF(64). The rate of this code has been adjusted so that the bandwidth is the same as used in the 64-ary system. For the second system, each channel symbol represents five bits and each code symbol represents six bits. So, six channel symbols are processed to obtain five code symbols.

No 16-ary system exists to compare to these systems because a code of the correct rate does not exist. However, a system using 8-ary signaling can be found which uses a (64,15) extended RS code which can correct twenty-four 64-ary symbols. Here, two channel symbols are sent for every code symbol.

A third type of system using 8-ary signaling can be compared to the above systems that uses a code with 8-ary code symbols instead of 64-ary symbols. This system can send twice as many code symbols since code symbols are the same as channel symbols. Thus a (128,30) 8-ary code can be used. This code is not a Reed-Solomon code, so another type of code, such as an extended nonbinary cyclic code, is necessary for implementation.

4.4 Error Control Performance

4.4.1 Errors Only Decoding

A set of equivalent systems is compared based on the resulting probability of bit error. An (n,k) code has an associated minimum distance, $d_{\min}$, which determines the number of errors the code can correct. A code with distance $d_{\min}$ can correct t errors, where

$$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor . \tag{4.5}$$

If a code uses a bounded distance decoder it will correct any received word that has t or fewer errors. If the received word has more than t errors then a decoder error (the word is decoded into an incorrect codeword) or a decoder failure occurs. Assuming the system cannot retransmit decoder failures, a received word which caused a decoder failure will not have any errors corrected. Thus, the probability of word error is the probability that more than t code symbols are in error,

$$P_w = \sum_{i=t+1}^{n} \binom{n}{i} p_e^i (1-p_e)^{n-i}, \tag{4.6}$$

where the probability that a code symbol was received in error, p_e, was given in Section 4.1, for the case where $M = q$ or there is an integer number of channel symbol in each code symbol ($\log_2 q / \log_2 M$ is an integer).

If $\log_2 q / \log_2 M$ is not an integer, then a code symbol will contain fractional parts of channel symbols. The performance of the system will depend on how the channel symbols are divided in the code symbols. For example, if $q = 64$ and $M = 32$ then a code symbol contains six bits and a channel symbol contains five bits. One way to divide the channel symbols is shown in Figure 4.1. The first code symbol contains one channel symbol and one bit from a second channel symbol. The second code symbol contains the remaining four bits from the second channel symbol plus two bits from the third channel symbol. It takes six channel symbols to transmit five code symbols.

For this transmission pattern each of the five code symbols will have a different probability of symbol error due to the different combination of channel symbols. Code symbol 1 will be correct if channel symbol 1 is correct and either channel symbol 2 is correct, or if channel symbol 2 is in error, then the first bit of the erroneous channel symbol 2 is correct. Thus,

$$p_{e_1} = 1 - (1 - p_{ce})(1 - \frac{16}{31} p_{ce}),$$

where p_{e_1} is the probability that code symbol 1 is in error, and p_{ce} is the probability of a channel symbol error. In general for a $q = 2^m$-ary code symbol that contain parts of two channel symbols,

$$p_{e_i} = 1 - (1 - p_{ce_i})(1 - p_{ce_{m-i}}), \tag{4.7}$$

where p_{ce_i} is the probability that at least one of the first i bits of a 2^b channel symbol is in error. This probability is given by

$$p_{ce_i} = \frac{2^b - 2^{b-i}}{2^b - 1} p_{ce}, \tag{4.8}$$

where p_{ce} is the probability that a channel symbol is received in error.

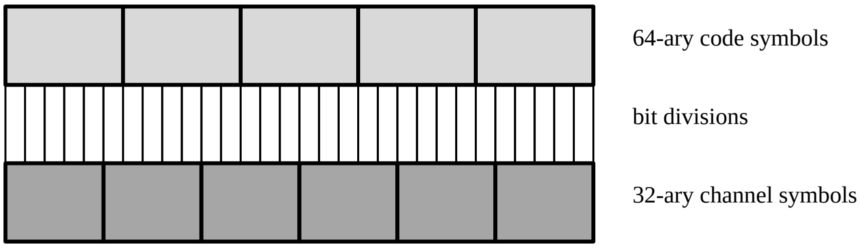

Figure 4.1 Example of Channel/Code Symbol Division

Another possible transmission pattern is shown in Figure 4.2. Here, each code symbol contains one complete channel symbol and one bit from another channel symbol. Using this transmission pattern each code symbol will have the same probability of error, which is the same as the probability of error for the first code symbol in the previous transmission pattern. Theorem 1 shows that the second transmission pattern, where the code symbols contain as many whole channel symbols as possible and remaining bits are from a single channel symbol, gives the best error performance.

Theorem 1 *The optimum transmission pattern for a $q = 2^m$-ary code symbol using $M = 2^b$-ary channel symbols is to use the largest number of channel symbols along with the remaining equivalent bits from a single channel symbol.*

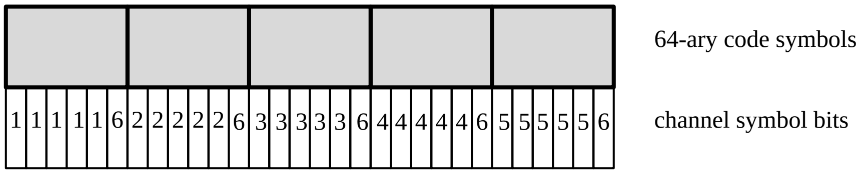

Figure 4.2 Optimal Channel/Code Symbol Division

Proof: Assume $2b < m$. If $2b \geq m$ each code symbol will contain a number of complete channel symbols and the same results hold. Let p_{e_i} be the probability of a code symbol error given that the code symbol contains i bits from one channel symbol, and $m-i$ bits from another channel symbol. Thus $p_{e_i} = p_{e_{m-i}}$ and only the case where $i \geq (m+1)/2$ must be considered.

Lemma 1 $p_{ce_i} \leq p_{ce_{i+1}}$.

Proof:

$$\begin{aligned}
p_{e_i} - p_{e_{i-1}} &= 1 - (1 - p_{ce_i})(1 - p_{ce_{m-i}}) - 1 + (1 - p_{ce_{i-1}})(1 - p_{ce_{m-i+1}}) \\
&= p_{ce_i} + p_{ce_{m-i}} - p_{ce_{i-1}} - p_{ce_{m-i+1}} - p_{ce_i} p_{ce_{m-i}} + p_{ce_{i-1}} p_{ce_{m-i+1}} \\
&= \frac{-2^{b-i} - 2^{b-m+i} + 2^{b-i+1} + 2^{b-m+i-1}}{2^b - 1} p_{ce} + \\
&\quad \frac{2^{2b-m+i} + 2^{2b-i} - 2^{2b-m+i-1} - 2^{2b-i+1}}{(2^b - 1)^2} p_{ce}^2 \\
&= \frac{-2^{b-i} + 2^{b-m+i-1}}{2^b - 1} p_{ce} \left[\frac{2^b p_{ce}}{2^b - 1} - 1 \right] \leq 0, \text{since } p_{ce} \leq \frac{2^b - 1}{2^b}.
\end{aligned}$$

Lemma 1 shows that whole channel symbols should be used in the makeup of code symbols.

Lemma 2 $1 - (1 - p_{ce_i})(1 - p_{ce_{m-i}}) \leq 1 - (1 - p_{ce_i})(1 - p_{ce_j})(1 - p_{ce_{m-i-j}})$.

Proof: Equivalently prove $(1-p_{ce_i}) \geq (1-p_{ce_{i-j}})(1-p_{ce_j})$ (replace $m-i$ by i).

$$(1-p_{ce_{i-j}})(1-p_{ce_j})-(1-p_{ce_i})$$

$$= \left(1-\frac{2^b-2^{b-i+j}}{2^b-1}p_{ce}\right)\left(1-\frac{2^b-2^{b-j}}{2^b-1}p_{ce}\right)-\left(1-\frac{2^b-2^{b-i}}{2^b-1}p_{ce}\right)$$

$$= \frac{\left(2^b-2^{b-i+j}\right)\left(2^b-2^{b-j}\right)}{\left(2^b-1\right)^2}p_{ce}^2-\frac{2^b-2^{b-i+j}-2^{b-j}+2^{b-i}}{2^b-1}p_{ce}$$

$$= \frac{\left(2^b-2^{b-i+j}\right)\left(2^b-2^{b-j}\right)}{2^b\left(2^b-1\right)}p_{ce}\left[\frac{2^b p_{ce}}{2^b-1}-1\right] \leq 0.$$

Lemma 2 shows that when part of a channel symbol is used to complete a code symbol it should be kept as large as possible. Theorem 1 follows.

Table 4.1 shows, for the cases presented in this dissertation, the resulting probability of code symbol error if the largest number of channel symbols is used with the remaining bits coming from the fewest number of channel symbols possible. In all of the cases except one, the code symbols within a particular code all have the same combination of channel symbols without having a large number of wasted channel symbols. If only one codeword is sent, a fraction of a channel symbol could be left over. If multiple codewords are sent, then all parts of the channel symbols can be used. In the case of $m = 5$ and $b = 3$ two different transmission patterns for the code symbols must be used in order to get the best possible results with the fewest number of channel symbols. Two-thirds of the code symbols are made of one channel symbol and two bits from another channel symbol, while the remaining one-third of the code symbols will have one complete channel symbol and two bits from two different channel symbols. These two possible code symbols will have different code symbol error probabilities.

Table 4.1 Probability of Code Symbol Error

Code Symbol Size m	Channel Symbol Size b	Pr(Code Symbol Error) p_e
6	5	$p_e = 1-(1-p_{ce})(1-\frac{16}{31}p_{ce})$
6	4	$p_e = 1-(1-p_{ce})(1-\frac{12}{15}p_{ce})$
6	3	$p_e = 1-(1-p_{ce})^2$
5	4	$p_e = 1-(1-p_{ce})(1-\frac{8}{15}p_{ce})$
5	3	$p_{e_3} = 1-(1-p_{ce})(1-\frac{6}{7}p_{ce})$ $p_{e_1} = 1-(1-p_{ce})(1-\frac{4}{7}p_{ce})^2$
4	3	$p_e = 1-(1-p_{ce})(1-\frac{4}{7}p_{ce})$

In all of the above cases except the $m = 5$, $b = 3$ case, all of the code symbols in the codeword have the same probability of symbol error. Thus, Equation 4.6 can be used directly for these cases. For the $m = 5$, $b = 3$ case, Equation 4.6 must be modified to account for two different code symbol error probabilities. Assuming twenty-two code symbols are made of one channel symbol and two bits from another channel symbol, and the other ten code symbols in the codeword are made from one complete channel symbol and one bit from each of two different channel symbols, the resulting codeword error probability is

$$P_{w_{m=5,b=3}} = \sum_{j=t+1}^{n} \sum_{i=i^*}^{j^*} \binom{22}{i} p_{e_3}^{i}(1-p_{e_3})^{22-i} \binom{10}{j-i} p_{e_1}^{j-i}(1-p_{e_1})^{10-j+i}, \quad (4.9)$$

where $i^* = \max(0, j-10)$ and $j^* = \min(j, 22)$.

The most likely event causing a received word to be decoded in error, is that $t+1$ of the code symbols are received in error. The receiver decodes this word to the closest codeword by introducing t additional errors for a total of $2t+1 = d_{\min}$ errors. Thus, an estimate of the decoded code symbol error is

$$P_e = \left(\frac{d_{\min}}{n}\right) P_w. \tag{4.10}$$

If all code symbol errors are equally likely, then, from [17], the probability of a bit error is

$$P_b = P_e \left(\frac{q}{2(q-1)}\right), \tag{4.11}$$

where q is the size of the code symbol alphabet.

4.4.2 Errors and Erasures Decoding

To compare equivalent systems that employ errors and erasures decoding, the probability of codeword error must be found. This probability is a function of the code symbol probability of error, the code symbol probability of erasure, and the error correction capability of the code.

The probability of a code symbol error or erasure is determined by the probability of a channel symbol error or erasure. The ratio threshold test (RTT) presented in Section 2.2.1 is used to determine which channel symbols to erase. From [8] the probability of a channel symbol error for a Rayleigh fading channel is

$$p_{ce} = \sum_{i=0}^{M-2} \binom{M-1}{i+1} (-1)^i \left(\frac{(i+1)\gamma^2}{(i\gamma^2+1)}\right) \left(\frac{1}{(i+1)\gamma^2 + 1 + E_s/N_0(i\gamma^2+1)}\right), \tag{4.12}$$

and the probability of channel symbol erasure is

$$p_{cs} = \sum_{i=0}^{M-2} \binom{M-1}{i+1} (-1)^i \frac{1}{(E_s/N_0)(i+1)\gamma^2 + (i+1)\gamma^2 + 1} - p_{ce}, \tag{4.13}$$

where γ is the RTT threshold and E_s/N_0 is the symbol energy to noise ratio. A code symbol is erased if one or more of its channel symbols are erased. Thus, the probability of a code symbol erasure is

$$p_s = 1 - (1 - p_{cs})^{\lfloor m/b \rfloor}. \tag{4.14}$$

A code symbol is in error if one or more of its channel symbols are in error and no channel symbols are erased. If $m = b$ or m/b is an integer, then the probability of code symbol error is

$$p_e = (1 - p_{cs})^{m/b} - (1 - p_{ce} - p_{cs})^{m/b}. \tag{4.15}$$

If m/b is not an integer, then the probability of code symbol error is dependant on the arrangement of the channel symbols within the code symbol. For all of the cases given in Table 4.1 except the $m = 5$, $b = 3$ case, the probability of symbol error is

$$p_e = (1 - p_{cs})^2 - (1 - p_{cs} - p_{ce_b})(1 - p_{cs} - p_{ce_{m-b}}) \tag{4.16}$$

Once the probabilities of code symbol error and code symbol erasure are found, a counting argument is used to find the probability of codeword error [8],

$$P_w = \sum_{i=0}^{n} \sum_{j=j_o(i)}^{n-i} \frac{n!}{i!j!(n-i-j)!} p_s^i p_e^j (1 - p_s - p_e)^{n-i-j}, \tag{4.17}$$

where $j_o(i) = \max(n - k + 1 - 2i, 0)$. The probability that a bit is decoded incorrectly is given in Equation 4.11.

4.4.3 Soft-Decision Decoding

To compare systems using soft-decision decoding, simulation techniques are used. The decoding algorithms described in Chapter 2 are modified to calculate reliabilities of code symbols from the reliabilities of channel symbols. The systems described in Chapter 2 use the same alphabet for the code and channel symbols. Since one channel symbol is the same as a code symbol, the metrics for the channel and code symbols are the same. With the systems considered in this chapter, one code symbol may consist of more than one channel symbol. Thus, the code symbol metrics are not always the same as the channel symbol metrics.

Consider a code symbol alphabet of size q, and a channel symbol alphabet of size M, where $q > M$. Each code symbol is represented by $\log_2 q$ bits, and each channel symbol is represented by $\log_2 M$ bits. Thus, a codeword is written as a series of $n \log_2 q$ bits which are transmitted as channel symbols of length $\log_2 M$ bits.

At the detector, the receiver's output vector $\mathbf{y}_i$ is processed into a hard-decision estimate of a channel symbol, r_i, and a reliability vector, $\mathbf{m}(\text{channel})_i = (m_{i,0}, m_{i,1}, \ldots, m_{i,M-1})$. The $n \log_2 q / \log_2 M$ channel symbol reliability vectors must be processed to produce reliabilities for the n codeword symbols.

If $\log_2 q / \log_2 M$ is an integer, then an integer number of channel symbols represent each code symbol. The code symbol reliability vectors are the average of the channel symbol metrics for the channel symbols in the code symbols. For example, if a 64-ary code is used with 8-ary signaling, two channel symbols represent one code symbol. If $m(\text{code})_{i,j}$ is the metric of the j-th symbol in the i-th code symbol location, and $m(\text{channel})_{r,s}$ is the metric of the s-th channel symbol sent in the r-th location, then the code metrics are found as follows:

$$m(\text{code})_{0,0} = \tfrac{1}{2}\left(m(\text{channel})_{0,0} + m(\text{channel})_{1,0}\right)$$

$$m(\text{code})_{0,1} = \tfrac{1}{2}\left(m(\text{channel})_{0,0} + m(\text{channel})_{1,1}\right)$$

$$\vdots$$

If $\log_2 q / \log_2 M$ is not an integer, then code symbols contain fractions of channel symbols. For example, if a 64-ary code is used with 32-ary signaling, the channel symbols are divided over the code symbols as shown in Figure 4.1. Therefore, a single channel metric may affect two code metrics. In the example shown in Figure 4.1 the first code symbol metric will depend on the first channel symbol metric, and all possibilities of obtaining a one or a zero in the most significant bit with the second channel symbol. For the zero-th code symbol metric, the zero-th channel symbol metric is needed for location 0 and all channel symbol metrics which have a first bit of zero, namely channel symbols 0-15 at channel location 1. The channel symbol metrics are weighted to reflect the portion of the code symbol that is made up of that channel symbol. One method of combining the channel symbol metrics is to take the maximum metric of the channel set. Thus, the code metrics are determined as follows:

$$m(\text{code})_{0,0} = \tfrac{5}{6}\left(m(\text{channel})_{0,0}\right) + \tfrac{1}{6}\left(\max_{i=0,\ldots,15}(m(\text{channel})_{1,i})\right)$$

$$m(\text{code})_{0,1} = \tfrac{5}{6}\left(m(\text{channel})_{0,0}\right) + \tfrac{1}{6}\left(\max_{i=16,\ldots,31}(m(\text{channel})_{1,i})\right)$$

$$m(\text{code})_{0,2} = \tfrac{5}{6}\left(m(\text{channel})_{0,1}\right) + \tfrac{1}{6}\left(\max_{i=0,\ldots,15}(m(\text{channel})_{1,i})\right)$$

$$\vdots$$

$$m(\text{code})_{1,0} = \tfrac{4}{6}\left(\max_{i=0,16} m(\text{channel})_{1,i}\right) + \tfrac{2}{6}\left(\max_{i=0,\ldots,7}(m(\text{channel})_{1,i})\right)$$

$$m(\text{code})_{1,1} = \tfrac{4}{6}\left(\max_{i=0,16} m(\text{channel})_{1,i}\right) + \tfrac{2}{6}\left(\max_{i=8,\ldots,16}(m(\text{channel})_{1,i})\right)$$

$$\vdots$$

4.5 Performance Results

Three types of codes are used for the results presented in this section, extended Reed-Solomon codes, BCH codes, and nonbinary cyclic or shortened cyclic codes. The minimum distance for the RS codes is $d_{\min} = n - k + 1$, and for the BCH codes is $d_{\min} = \delta$, the design distance of the code. For the cyclic and non-cyclic codes the weight distributions must be found be a search procedure

4.5.1 Errors Only Decoding

Figures 4.3-4.6 show four different sets of comparable systems. Each set contains an original system that uses M_1-ary orthogonal signaling and a (M_1, k) extended RS code. The other systems in the set use M_2-ary signaling and a code that has an appropriate rate to keep the bandwidth utilization constant. The systems used for these figures employ hard-decision decoding. The bit energy to noise ratio has been adjusted to account for different signal set sizes and for different code rates. The systems use noncoherent receivers and communication is over a Rayleigh fading channel.

Figure 4.3 shows the performance of the set of systems described in Section 4.3. The extended RS (64,60) code corrects two code symbol errors, while the extended RS (64,36) corrects fourteen code symbol errors. From this gain in code performance, the second system can overcome the lower probability of non-coded channel symbol error of the 64-ary system. However, going to the extended RS (64,15), 8-ary system offers improvement over the 32-ary system only for large bit energy to noise ratios even though the extended RS (64,15) code can correct twenty-four code symbol errors. For the low bit energy to noise ratios, the code cannot overcome the higher probability of uncoded symbol error caused by the lower energy per bit due to the lower modulation rate and, most significantly, the energy loss encountered because of the code rate. As shown, the 8-ary BCH (128,30) does not perform as well as the RS (64,15). The BCH code used here is a weak code; it corrects errors up to its design distance only when BCH hard-decision decoding is used. Other 8-ary length 128 codes can be found to improve this performance, such as a cyclic code. However, in order to use this code, an efficient method of performing hard-decision decoding would have to be found.

Figure 4.4 contains results from a set of systems based on an 64-ary system using an extended RS (64,40) code. The (64,40) code is more powerful than the (64,60) code used for Figure 4.3, so the benefit of going to lower modulation sizes and better codes does not occur until higher bit energy-to-noise ratios then for the systems used for Figure 4.3. Notice for the 64-ary system using the (64,40) code, 32-ary, 16-ary, and 8-ary systems can all be found to compare to the 64-ary system. The best improvement for the system is obtained by going to a 32-ary system with an extended RS (64,24) code. Further reductions in the signal set size do not provide additional improvements except at high bit energy to noise ratios.

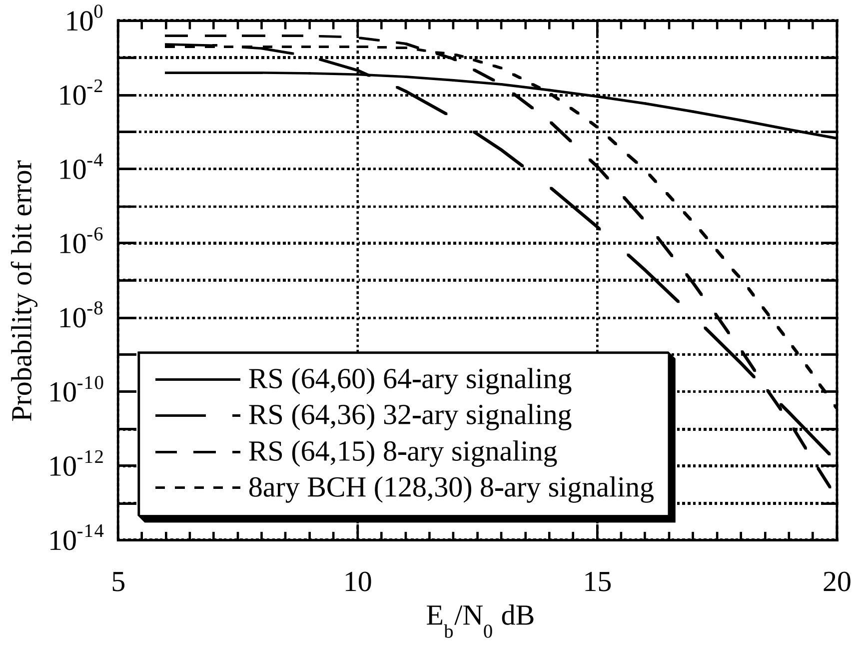

Figure 4.3 Systems based on 64-ary Orthogonal Signaling, RS (64,60) Using Hard-Decision Decoding

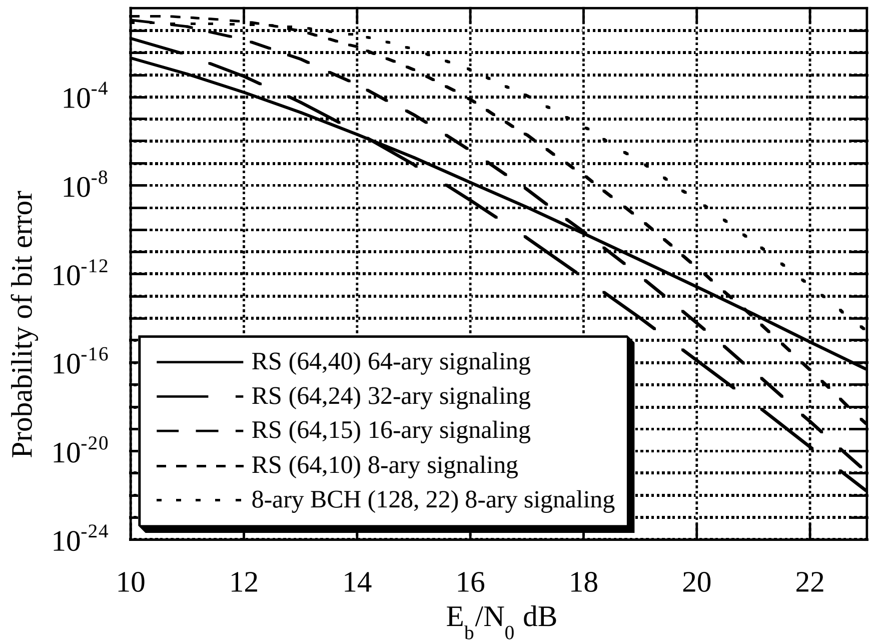

Figure 4.4 Systems based on 64-ary Orthogonal Signaling, RS (64,40) Using Hard-Decision Decoding

Figure 4.5 shows the performance of a set of systems based on a 32-ary system with an extended RS (32,24) code. As the bit energy to noise ratio increases, the second system, a 16-ary system using an extended RS(32,15) code, performs better than the 32-ary system. As the bit energy to noise ratio is increased further, the third system, using 8-ary signaling and an extended RS (32,10) code performs better than the 32-ary system and the 16-ary system.

The curves in Figure 4.6 are based on a 16-ary system using an extended RS (16,12) double error correcting code. Most of the improvement occurs from using the 8-ary system, but the 4-ary systems will perform better than the 8-ary system at higher bit energy to noise ratios. The 4-ary system using a 4-ary extended BCH (32,12) code performs as well as the 4-ary system using an extended RS (16,6) code. Additional improvements could be gained if a better 4-ary length 32 code is used.

4.5.2 Errors and Erasures Decoding

The systems used for the results in Section 4.4.3 are used again; however, these systems employ errors and erasures decoding using the RTT to generate the erasures. The use of errors and erasures decoding has provided better error correction performance, as is evident from the curves shown in Figures 4.7-4.10.

In all four figures the resulting probability of error curves for the modified systems show more improvement than for the original system. The performance gain in going to a smaller constellation size with a more powerful curve occurs at lower bit energy-to-noise ratios when using errors and erasures decoding than when using hard-decision decoding. For example, the RS (64,36) 32-ary system has a probability of bit error curve shown in Figure 4.3, that crosses the probability of error curve of the RS (64,60) 64-ary system at about 8 dB when using hard-decision decoding, but the probability of error curve, shown in Figure 4.7, crosses at approximately 7 dB when using errors and erasures decoding. Similar results are seen with the rest of the systems.

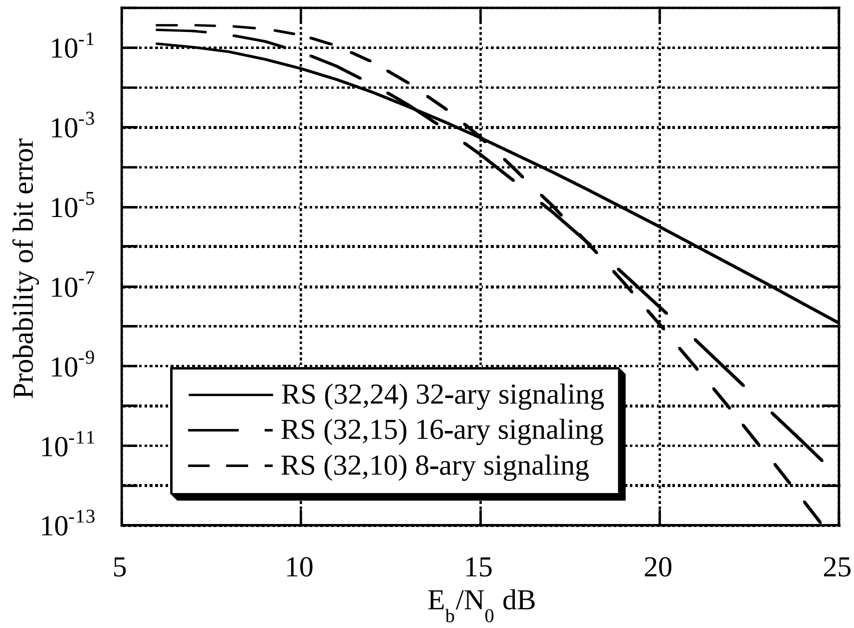

Figure 4.5 Systems based on 32-ary Orthogonal Signaling, RS (32,24) Using Hard-Decision Decoding

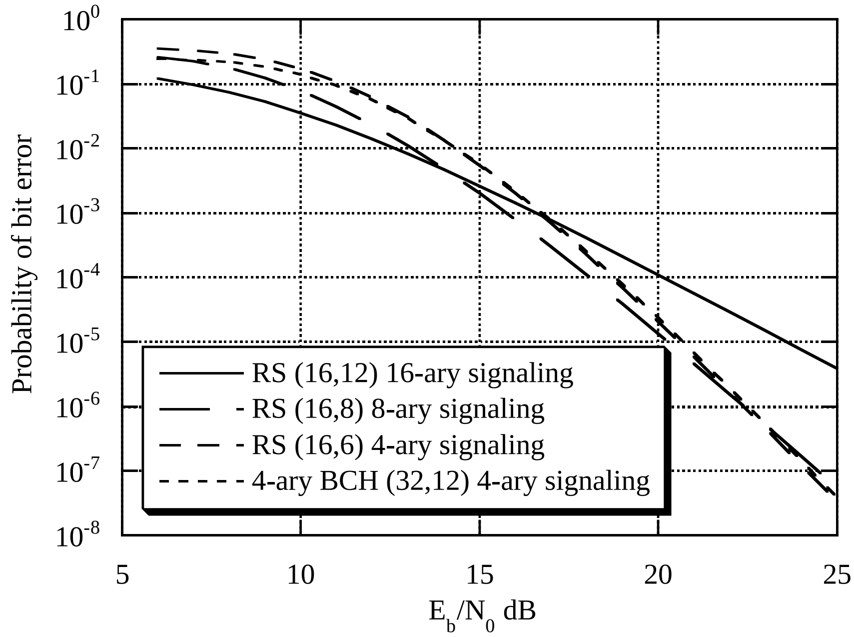

Figure 4.6 Systems based on 16-ary Orthogonal Signaling, RS (16,12) Using Hard-Decision Decoding

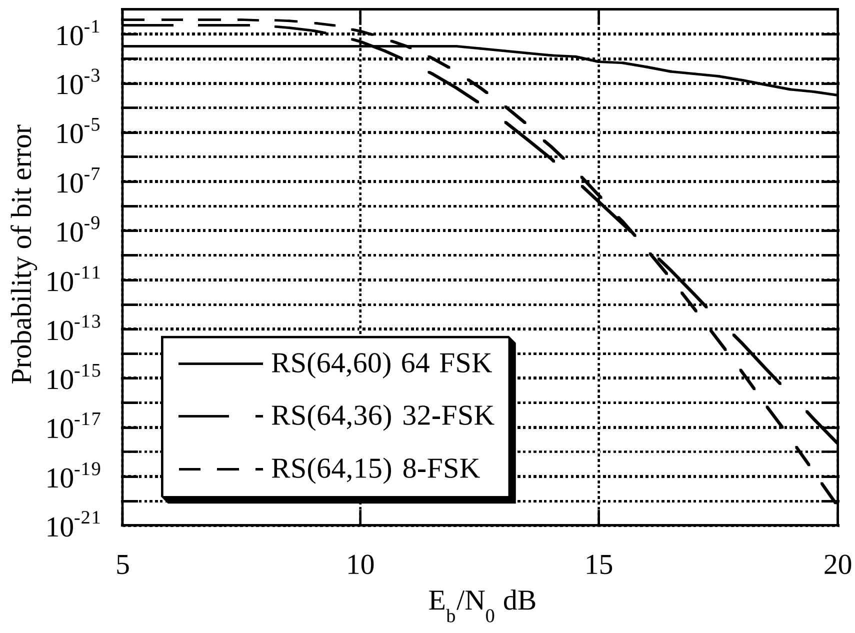

Figure 4.7 Systems based on 64-ary Orthogonal Signaling, RS (64,60) Using Errors and Erasures Decoding

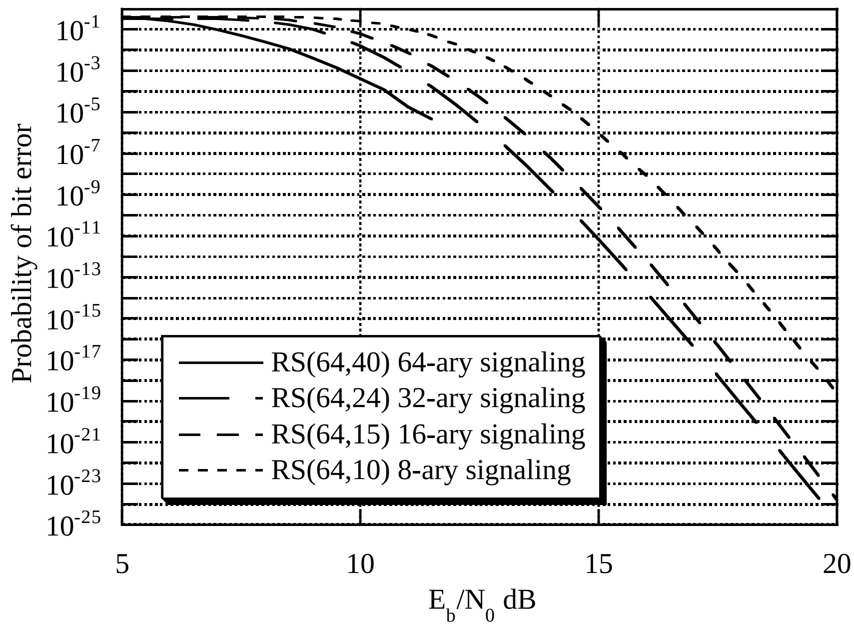

Figure 4.8 Systems based on 64-ary Orthogonal Signaling, RS (64,40) Using Errors and Erasures Decoding

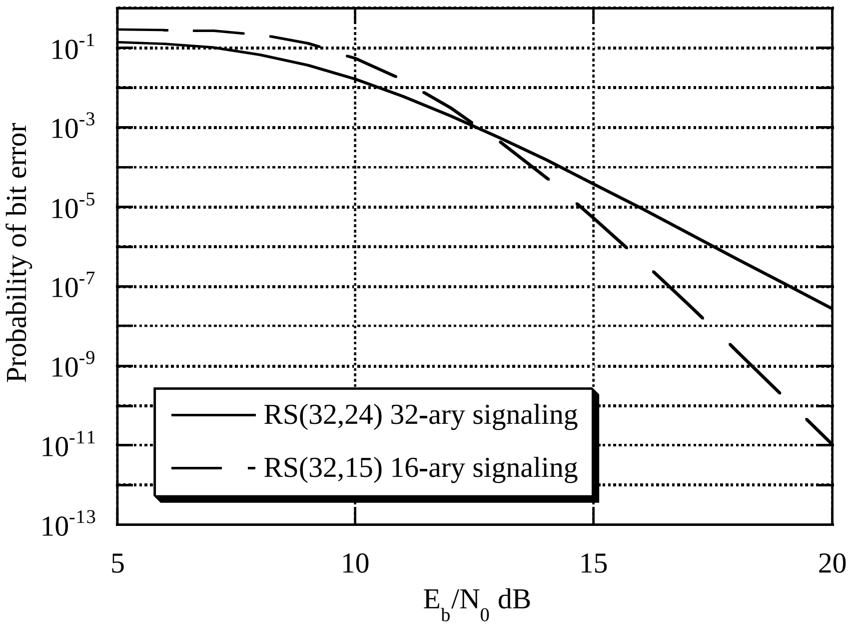

Figure 4.9 Systems based on 32-ary Orthogonal Signaling, RS (32,24) Using Errors and Erasures Decoding

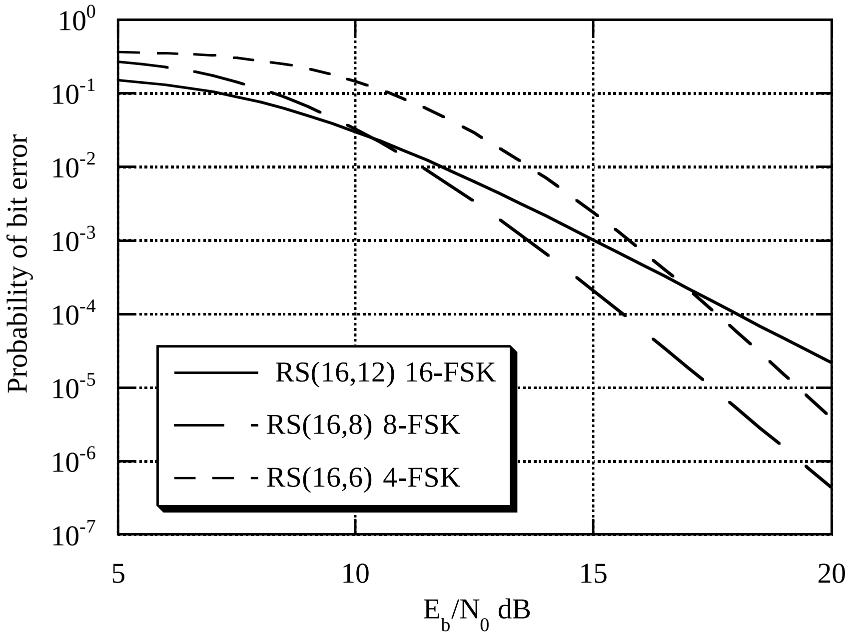

Figure 4.10 Systems based on 16-ary Orthogonal Signaling, RS (16,12) Using Errors and Erasures Decoding

4.5.3 Soft-Decision Decoding

The results in this section are obtained from simulations of the set of systems based on the system using 64-ary signaling and an extended RS(64,60) code and the set based on the system using 16-ary signaling and an extended RS(16,12) using soft-decision decoding. All of the systems use the extended error magnitude matrix for decoding and allow the decoder to correct single errors in the k data locations. To increase performance, all possible symbol values for these k locations are tried as needed.

The probability of bit error achieved when using systems based on the RS(64,60) systems are shown in Figure 4.11. The error probabilities using the tree algorithm are all lower than with error and erasure decoding. The use of soft-decision decoding has the most effect on the more powerful codes. The probability of bit error curves for the modified systems cross the probability of bit error curve for the RS (64,60) 64-ary system at lower bit energy to noise ratio than when errors and erasures decoding was used. For this system a shortened cyclic (128,31) 8-ary code is used. This system has a lower probability of bit error than any of the rest of the systems in the set. The long code provides better error correction performance than the shorter RS(64,15) code.

The results of the systems based on 16-ary signaling and a RS(16,12) code are shown in Figure 4.12. A 4-ary (31,11) cyclic code is used in a 4-ary system. This system does not perform as well as the RS(64,15) 4-ary system. The cyclic code was chosen at random since the minimum distances for the (31,11) codes are not known. It is possible that a better code can be found.

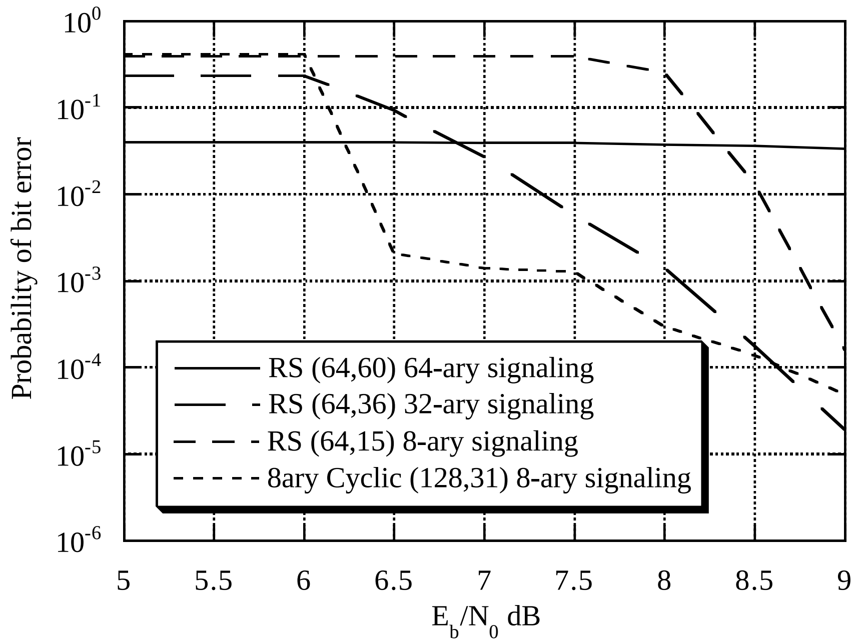

Figure 4.11 Systems based on 64-ary Orthogonal Signaling, RS (64,60) Using Soft-Decision Decoding

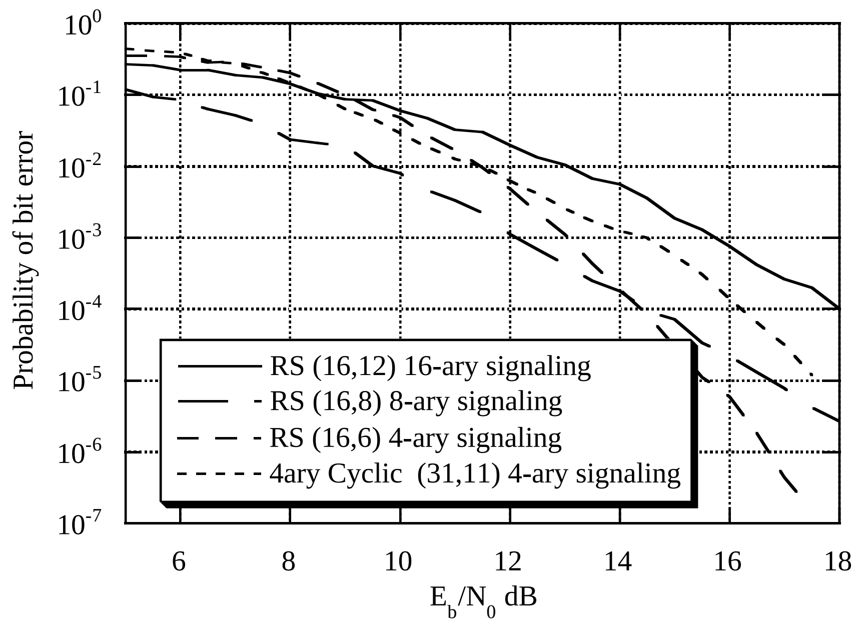

Figure 4.12 Systems based on 16-ary Orthogonal Signaling, RS (16,12) Using Soft-Decision Decoding

CHAPTER 5

CONCLUSTIONS

Cyclic codes are widely used because of their algebraic structure that makes for efficient decoding algorithms. Shortened cyclic codes can be decoded using the same algorithms as cyclic codes. hard-decision decoding, such as errors only decoding and errors and erasures decoding, are efficient, but do not use all of the information available at the receiver. Better error performance is gained by using soft-decision decoding to use the reliability information. In this dissertation several soft-decision decoding algorithms are presented to decode Reed-Solomon codes. These new soft-decision decoding algorithms provide efficient decoding and better error correction performance than previous decoding techniques.

The error correction capabilities of these algorithms are compared to existing hard-decision and soft-decision decoding algorithms. When these algorithms are simulated for a Rayleigh fading channel, the new soft-decision algorithms out perform the hard-decision algorithms significantly, even when the number of computations is limited. The new sequential decoding algorithm also has better performance than GMD decoding while requiring fewer field computations, but additional memory requirements are needed.

Since Reed-Solomon codes exist only for specific code lengths, non-binary cyclic codes are examined to find q-ary codes of arbitrary lengths with good error correcting capabilities. General cyclic and shortened cyclic codes are presented and examples of these codes compared to BCH codes are given. Since these codes do not have all of the algebraic structure of the Reed-Solomon codes or BCH codes, the soft-decision decoding algorithms are modified to work for these codes. The modified extended error magnitude matrix decoding algorithm is only slightly more complex than the original algorithm for RS codes, and once the matrix is generated, decoding is identical. Due to the structure of the codes, it is not guaranteed that the received symbols that are most likely to be in error are erased, but this has a minimal effect on performance. The extended error erasure matrix decoding method provides soft-decision decoding up to the actual distance of the cyclic code. If hard-decision decoding is used, BCH decoding techniques would are used which only correct up to the BCH design distance of the code.

The soft-decision decoding algorithms can be used in systems which use M-ary orthogonal signaling. Equivalent systems can be created which have the same bandwidth and energy requirements, but use different signal constellation sizes. By reducing the signal constellation size, a code of lower rate is used while keeping bandwidth requirements constant. The lower rate code can correct more errors which gives a better system performance. If soft-decision decoding is used, then systems can be developed to use cyclic codes and further increase performance.

REFERENCES

1. S. B. Wicker. *Error Control Systems for Digital Communication and Storage.* Prentice Hall, 1995.

2. S. K. Shin and P. Sweeney. Soft decision decoding of Reed-Solomon codes using trellis methods. *IEE Proceedings on Communications*, 141(5):303–308, October 1994.

3. J. J. Komo and L. L. Joiner. Fast error magnitude evaluations for Reed-Solomon codes. In *Proc. 1995 IEEE International Symposium on Information Theory*, page 416, September 1995.

4. S. Lin and D. J. Costello, Jr. *Error Control Coding: Fundamentals and Applications.* Prentice Hall, 1983.

5. G. D. Forney, Jr. On decoding BCH codes. *IEEE Transactions on Information Theory*, 11(4):549–557, October 1965.

6. L. L. Joiner and J. J. Komo. Time domain decoding of extended Reed-Solomon codes. In *Proc. 1996 IEEE Southeast Con*, pages 238–241, April 1996.

7. A. J. Viterbi. A robust ratio-threshold technique to mitigate tone and partial band jamming in coded MFSK systems. In *IEEE Military Communication Conference*, pages 22.4.1–22.4.5, October 1982.

8. C. W. Baum. *Decision-Theoretic Techniques for the Development and Use of Side Information in Frequency-hop Radio Receivers.* PhD thesis, University of Illinois at Urbana-Champaign, 1992.

9. G. D. Forney, Jr. Generalized minimum distance decoding. *IEEE Transactions on Information Theory*, 12(2):125–131, April 1966.

10. D. J. Taipale and M. B. Pursley. An improvement to generalized-minimum-distance decoding. *IEEE Transactions on Information Theory*, 37(1):167–172, January 1991.

11. E. R. Berlekamp. *Algebraic Coding Theory.* McGraw-Hill, 1968.

12. D. J. Taipale and M. J. Seo. An efficient soft-decision Reed-Solomon decoding algorithm. *IEEE Transactions on Information Theory*, 40(4):1130–1139, 1994.

13. U. K. Sorger. A new Reed-Solomon code decoding algorithm based on newton's interpolation. *IEEE Transactions on Information Theory*, 39(2):358–365, March 1993.

14. M. Morii and M. Kasahara. Generalized key-equation of remainder decoding algorithm for Reed-Solomon codes. *IEEE Transactions on Information Theory*, 38(6):1801–1807, November 1992.

15. E. R. Berlekamp. Bounded distance + 1 soft-decision Reed-Solomon decoding. *IEEE Transactions on Information Theory*, 42(3):704–720, May 1996.

16. W. Reid, III, J. J. Komo, and L. L. Joiner. Maximum error magnitude decoding of the Golay [23,12,7] code. In *Proc. 1998 IEEE International Symposium on Information Theory*, page 219, August 1998.

17. M. B. Pursley. *Introduction to Digital Communications.* Addison-Wesley, 1999.

18. J. G. Proakis. *Digital Communications.* McGraw-Hill, 1995.