AN ABSTRACT OF THE THESIS OF

Robert Stephen Cunningham for the degree of Master of Science in Computer Science presented on March 1, 1982.

Title: On Asymmetric Error-Correcting Codes

Redacted for Privacy

Abstract approved: ______________________
Dr. Bella Bose

Historically, coding theory has dealt with binary codes correcting symmetric errors, in which errors are made in both 0 and 1 bits with equal likelihood. Within the past ten years, some study has been made of asymmetric codes, under the assumption that the only errors which occur are errors in which 1 becomes 0. This thesis continues this study.

We first examine systematic asymmetric codes, binary codes for which information and check portions are in distinct bit fields. This is a new area of study in coding theory. We establish that systematic asymmetric codes can have higher information rates than systematic symmetric codes, but not too much higher. We also give a construction for building systematic codes from smaller ones, with necessary and sufficient conditions for the codes so built to be systematic asymmetric codes.

Finally, we examine Constantin-Rao codes and their extension to multiple asymmetric error correction. We show that such codes are not systematic and describe conditions under which they are closed under complements. We also show that the multiple asymmetric error correcting codes can have higher information rates than their symmetric counterparts.

On Asymmetric Error-Correcting Codes

by

Robert Stephen Cunningham

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of
Master of Science

Commencement June 1982

APPROVED:

Redacted for Privacy

Assistant Professor of Computer Science in charge of major

Redacted for Privacy

Chairman of Department of Computer Science

Redacted for Privacy

Dean of Graduate School

Date thesis is presented: March 1, 1982

Typed by Robert Stephen Cunningham

ACKNOWLEDGEMENT

This work was supported in part by
the National Science Foundation under
Grant No. SPI-8013280.

TABLE OF CONTENTS

I. Introduction 1

II. Systematic Asymmetric Codes 4

III. The Constantin-Rao Codes 18

IV. A Code Correcting Multiple Asymmetric Errors 25

V. Bibliography 34

VI. Appendix 36

List of Tables

Table 1: Code Sizes for Given Code Lengths 30

Table 2: Code Sizes for Given Code Lengths 31

Table 3: Element with the Maximal Preimage 32

Table 4: Closure Under Complements 33

On Asymmetric Error-Correcting Codes

I. Introduction

Accurate, reliable computer systems need more than correct arithmetic and logic to produce valid results. Particularly now, as computing and telecommunications become more and more closely related, efficient methods of ensuring data accuracy are of equal importance. This is the ground covered by coding theory: to devise accurate and efficient methods of storing, retrieving, and transmitting data.

The methods of coding theory involve adding a certain amount of redundancy to the information content of data so that should certain errors occur in the data before it is used, the redundancy will allow the error to be noticed or corrected before use. The redundancy can be added in many ways, but combinatoric or algebraic schemes are usually used. Since encoding and decoding of data takes time, one goal of coding theory is to find simple techniques which allow the coding to be done in parallel with the main processing activities.

Since coding methods are designed to be used with actual computing equipment, shifts in technology lead to new aspects of coding theory. Most of the results in coding theory have assumed equal probability of the two kinds of binary errors: 0 becomes 1, or 1 becomes 0. This is appropriate to electrical transmissions in wire or storage in magnetized cores. However, light transmission in glass cables and storage in semiconductor memories show much more frequent errors where 1 becomes 0 than conversely, which raises new questions for codes. Codes which are designed to detect and correct errors of this sort are called <u>asymmetric</u> codes, and in this thesis we attempt to deal with these.

This work has two primary directions. The first is to examine systematic codes, that is, codes in which the information and check (redundant) portions of the code are stored in separate portions of the code word. Such codes allow parallel decoding, with the information part of the code available for immediate use while checking is being done, and allowing restart of computation with the corrected data if an error is found. Systematic codes are studied for asymmetric error correcting. Such systematic codes are a very new topic for coding theory, for while asymmetric codes have been studied for about ten years, they have not been considered from the systematic point of view. We establish that systematic asymmetric codes can have information rates exceeding the best known systematic symmetric codes, but not by a great deal.

The second direction of this work is in extending the results on asymmetric codes defined by certain functions from binary vectors into Abelian groups or Galois fields. Here our work is not definitive, but is limited to filling in a few gaps and modifying others' constructions to obtain results casting additional light on these codes.

II. Systematic Asymmetric Codes

In this chapter we shall set out the basic definitions and background results on asymmetric codes, and shall then outline some general results on systematic asymmetric codes. A particular case is examined in detail and exact delineations of the possible codes are made for this case. Finally, we define a structure which gives necessary and sufficient conditions for constructions of such codes generally.

2.1. Definitions. Let C be a binary code with code words of length n.

2.1.1. C is a systematic code if there is a subset of k bits in the code indices so that the information portion of the code is contained in the k bits while the check portion of the code is contained in the remaining $n-k$ bits. Such a code is called a (n,k) code. The k information bits comprise the information symbol of the code word, while the check bits make up the check symbol.

2.1.2. C is an asymmetric code if C is capable of detecting or correcting errors in a channel in which errors where 0 becomes 1 are impossible (such a channel is called the Z-channel). We will call C a t-asymmetric code if C can correct t errors in the Z-channel.

2.1.3. For any two words a, b of C we let N(a,b) be the number of bit positions in which a contains 1 and b contains 0.

2.1.4. The asymmetric distance between two words a, b of C, denoted by $D_a(a,b)$, is defined by $D_a(a,b) = \max(N(a,b), N(b,a))$. Further, the asymmetric distance of a code C is defined to be $D_a(C) = \min\{D_a(a,b)$ for a, b in C$\}$.

2.1.5. For words a, b in C, we say a dominates b, and write a>>b, if b can be made from a by one or more changes of 1-bits to 0-bits.

2.1.6. If C is a systematic (n,k) code, the information rate R of the code is defined to be k/n, the proportion of the code which is used for actual information.

2.1.7. For a word a of a code C, the weight of a is the number of 1-bits in the word.

In the same way that codes may be shown to correct symmetric errors by considering the standard Hamming distance of the code [7], there is a relation between asymmetric distance and correction of asymmetric errors. This relation is given by the following theorem.

2.2. Theorem [8]. A binary code C is a t-asymmetric code if and only if $D_a(C) \geq t+1$.

Proof (sketch). For any code word X of C, let $S(X)$ be the set of vectors obtainable from X by replacing 1's by 0's in t or fewer places. For any two code words X and Y in C, the code can correct t asymmetric errors precisely if $S(X)$ and $S(Y)$ are disjoint. However, it is straightforward to see that this condition holds precisely if one of $N(X,Y)$ or $N(Y,X)$ is at least $t+1$.

Consider now optimal codes with r check bits. Since any code which can correct single symmetric errors can also correct single asymmetric errors, such codes are 1-asymmetric codes. Now the Hamming (n,k) codes with $n = 2^r - 1$ and $k = n - r$ are perfect codes and hence are optimal for their length. These are the best known symmetric single error correcting codes with r check bits; see for example [7], Table 5.4. This establishes the following floor on the size of possible systematic 1-asymmetric codes.

2.3. Proposition. For any $r > 0$, there is a systematic 1-asymmetric code of length $2^r - 1$ with r check bits.

On the other hand, an upper bound on the size of systematic 1-asymmetric codes is given by the following unpublished results of B. Bose.

2.4. Proposition (Bose). Any systematic 1-asymmetric code with k information bits must have at least $\log_2(k+1)$ check bits.

Proof. Consider the code words whose information symbols have weight 0 or 1. There are k+1 such words. If any two of these had the same check symbol they would have asymmetric distance 1, so the code could not correct single asymmetric errors by Theorem 2.2. In order to have the necessary k+1 distinct check symbols, at least $\log_2(k+1)$ check bits are needed.

Thus if $k \geq 2^m$, an systematic 1-asymmetric code with k information bits must have at least m+1 check bits. Since the Hamming codes with m+1 check bits are known to be optimal symmetric codes for $k < 2^{(m+1)} - (m+1)$, this shows that the Hamming codes are optimal systematic 1-asymmetric codes for $2^m \leq k < 2^{(m+1)} - (m+1)$. The next theorem, however, extends this range even farther.

2.5. Theorem (Bose). Let C be an systematic 1-asymmetric code with k information bits, where $2^m - m + 2 \leq k \leq 2^m - 1$ for $m \geq 3$. Then the number of check bits in C must be at least $m + 1$.

Proof. Assume by way of contradiction that C has m check bits. Note first that the code word with information symbol 0 cannot have all its check bits 1, for there would then be k words with weight-1 information symbols which could not have check symbols of weight m or $m-1$. This would leave $2^m - m - 1$ remaining check symbols, but $2^m - m - 1 < k$.

By an extension of this argument we have $2^m - m$ check symbols of weight unequal to $m-1$, and k information symbols of weight 1. Since $k > 2^m - m$, some code word in C must have a weight-1 information symbol and a check symbol of weight $m-1$; call this word w. Now consider all the weight-2 information symbols dominating the information symbol of w. There are $k-1$ such symbols, and they must have distinct check symbols. However, there are $m-1$ check symbols which may not be used--check symbols formed from the check symbol of w by changing a single 1 to 0--besides the check symbol of w itself. Thus there are $k-1$ information symbols and $2^m - m$ check symbols available, but $2^m - m < k - 1$. This contradiction established the theorem.

As a consequence, we see that the Hamming codes are optimal, even for correcting single asymmetric errors, for all cases except $k = 2^m - m$ and $k = 2^m - m + 1$. In these two cases, a Hamming code would have $r = m+1$ check bits, while a systematic 1-asymmetric code could have $r = m$ check bits. We will now turn our attention to these two cases with $r = 3$. Although this seems to be a very small case, no previous results of any kind are available and we are able to solve this particular problem completely. Recall that the results above show that the maximum length for a systematic 1-asymmetric code with 3 check bits is at least 7 and is less than 10. Thus our question for $r = 3$ becomes: are there any (8,5) or (9,6) systematic 1-asymmetric codes? We begin our answer by establishing the nonexistence of the (9,6) code.

2.6. Lemma. In a (9,6) systematic 1-asymmetric code, the information symbol 0 must have check symbol 0.

Proof. Assume this is false. Then the check symbol associated with the information symbol 0 could have weight 1, 2, or 3. We shall argue by cases.

Assume the check symbol has weight 3. Then the check symbols of weights 2 or 3 could not be used for any weight-one information symbols. Since there are 6 weight-one information symbols and only 4 check symbols of weight 0 or 1, two weight-one information symbols would have to have the same check symbol. This is clearly impossible.

Assume the check symbol has weight 2. Then no weight-one information symbol can have as check symbol either that check symbol or either of the two weight-one check symbols it dominates. This leaves 5 check symbols for the 6 weight-one information symbols, again an impossibility.

Finally, suppose the check symbol has weight one. Then no weight-one information symbol can use this check symbol or 0 as its check symbol. Since this leaves 6 remaining check symbols and none can be repeated among the 6 weight-one information symbols, one of these must have a weight-3 check symbol. Now consider the 5 weight-two information symbols which dominate this particular weight-one information symbol. As argued above, none of these can have check symbol of weight 2 or 3. Thus we have 4 check symbols available for these 5 information symbols, which again is impossible.

This establishes the lemma.

2.7. Lemma. In a (9,6) systematic 1-asymmetric code, there is a one-to-one correspondence between the weight-one information symbols and the check symbols having weights 1 or 2.

Proof. As was argued in the previous lemma, no weight-one information symbol can have a weight-3 check symbol, and by the lemma none can have check symbol 0. Since we have six remaining check symbols and six weight-one information symbols, and no two of these information symbols can have the same check symbol, the result is established.

2.8. Theorem. There exists no (9,6) systematic 1-asymmetric code.

Proof. Assume that such a code exists. By way of notation, call a code word c a [m,n]-word if the information symbol of c has weight m and the check symbol of c has weight n. Now by Lemma 2.6, there are 3 [1,2]-words in C. For each, there are 5 words with information symbol of weight 2 dominating its information symbol, and as in the proof of Lemma 2.5, there are 5 check symbols available for them. Thus each of the [1,2]-words must have a [2,0]-word whose information symbol dominates its information symbol. However, if the information symbol of a [2,0]-word dominates that of a [1,1]-word, an asymmetric distance of 1 results. Thus the information symbol of a [2,0]-word can only dominate that of a [1,2]-word. Thus there must be at least two [2,0] words in the code whose information symbols dominate the [1,2] words above. But this forces two [2,0] words to share a position in which each has a 1-bit, although two such words can have asymmetric distance only one. This contradicts the assumption of the existence of the (9,6) code.

We now turn our attention to (8,5) systematic 1-asymmetric codes. We have shown that such codes do exist, and have generated them by a rather simple-minded direct computation. An algorithm for this computation is presented below, and a Pascal program implementing the algorithm is presented in the Appendix.

2.9. Algorithm. This algorithm will compute and print all (n,k) systematic 1-asymmetric binary codes. Note that the work is really done in the subprogram, which uses a recursively backtracking process.

Data Structures: We have two arrays, each with 2^k rows. One array, INFO, represents the information symbols and has k columns, while the other, CHECK, represents the check symbols and has n - k columns.

Processing:

```
(A) Main Program
     (1)  load each row of INFO with the binary
          representation of the corresponding
          information symbol,
     (2)  invoke the procedure build_row below to
          determine the code's check symbols,
          starting with the first word.
(B) Procedure build_row(row); build check symbol for
     next code word.
        if row is larger than 2**k
           print out INFO and CHECK for the
           successful code
        else
           set check sum to 0
           while the check sum is less than 2**(n-k)
           do
              load the next row of CHECK with the
                 binary value of the check sum,
              increment the check sum,
              compute the asymmetric distance from
                 the new code word to all previous
                 code words,
              if the distance is at least 2
                 recursively invoke build_row with
                 parameter row+1,
           end-while
     end.
```

2.10. Example. Here we list four of the early (8,5) codes generated by this algorithm as expressed in the program sys85 listed in the Appendix. We have been unable to get the computer time needed for a full determination of all such codes; there are certainly several thousand of them. The codes are given in a highly compressed form, with the information and check symbols expressed by their decimal equivalents.

information	0	1	2	3	4	5	6	7	8	9	10	11
check 1	0	1	2	3	3	4	5	6	4	5	6	7
check 2	0	1	2	3	4	5	6	7	3	4	5	6
check 3	0	1	2	3	6	7	5	0	5	2	7	4
check 4	0	1	3	4	2	3	6	5	4	2	7	1

information	12	13	14	15	16	17	18	19	20	21	22	23
check 1	7	2	0	1	5	6	7	0	0	1	2	3
check 2	7	2	0	1	5	6	7	0	2	3	1	4
check 3	0	1	2	5	4	5	6	7	1	2	3	4
check 4	5	7	0	2	6	5	0	2	7	0	1	6

information	24	25	26	27	28	29	30	31
check 1	5	6	7	0	0	1	2	3
check 2	0	1	2	3	4	5	6	7
check 3	3	6	0	1	5	3	6	7
check 4	1	3	4	5	2	4	3	7

Having been successful in constructing (8,5) codes, we would like to extend our work to (16,12) and larger codes. We have not yet been able to do this. Direct computation as in the (8,5) case has not been possible because all our work has been done in a 16-bit minicomputer environment, where very large arrays are not addressable. However, the experience gained with the (8,5) case has allowed us to develop some results which will help us build these larger codes. Our results are framed in terms of _lattices_, which are partially-ordered sets in which each pair of elements have a least upper bound and greatest lower bound. The set of all binary vectors of length k forms a lattice, where a is greater than b if a>>b. The least upper bound of a and b is the logical join or _or_ of the vectors, while the greatest lower bound is the meet or _and_ of the vectors. We say that _a is a parent of b_ if a > b and there is no c with a > c and c > b, and that _a is a sibling of b_ if a and b share a common parent. In the lattice of binary vectors, a is a parent of b if a dominates b and the weight of a is one more than the weight of b.

Consider any (8,5) code and separate the information symbols into two parts: the two high-order bits and the three low-order bits. If the high-order bits are held constant, the low-order three bits and check bits form a systematic (6,3) 1-asymmetric code. We thus have four (6,3) codes labeled by the high-order two bits. Using the high-order bits to form a four-element lattice, we can then represent the (8,5) code as

2.11
$$\begin{tikzcd} & A \arrow[dl, dash] \arrow[dr, dash] & \\ B \arrow[dr, dash] & & C \arrow[dl, dash] \\ & D & \end{tikzcd}$$

where A, B, C, and D are the (6,3) codes associated with 11, 10, 01, and 00 respectively. Each of these has asymmetric distance at least two. We shall call A, B, C, and D the <u>suffix codes</u> and 11, 10, 01, and 00 the <u>prefix symbols</u>. In the examples in 2.10 above, then, the information symbols 0-7 give subcode A, 8-15 give subcode B, 16-23 subcode C, and 24-31 subcode D. As minimal-distance requirements among A, B, C, and D, then, we have the following:

(a) $N(A,B) \geq 1$ and $N(A,C) \geq 1$,
(b) $N(B,D) \geq 1$ and $N(C,D) \geq 1$,
(c) either $N(B,C) \geq 1$ or $N(C,B) \geq 1$.

Here, in saying that $N(A,B) \geq 1$ is a minimal-distance requirement we mean that $N(A,B) \geq 1$ or $N(B,A) \geq 2$. In fact, these conditions are precisely the conditions we see met by the codes in example 2.10, while some of the codes we found meet the stricter condition of A = D.

We are now able to give a similar characterization for systematic 1-asymmetric (n,k) codes. This should provide a tool for further studies in constructing systematic 1-asymmetric codes or in disproving their existence.

2.12. Theorem. Let C be a systematic (n,k) code, let t be any integer with $n > t > n-k$, and let $p = n-t$. For each of the 2^p prefix symbols a of length p, let $C(a)$ be the suffix code consisting of all code words in C whose prefix symbol is a. Then C is a 1-asymmetric code if and only if each suffix code $C(a)$ has $D_a(C(a)) \geq 2$ and the following requirements hold:

(a) if a and b are prefix symbols and a is a parent of b, then $N(C(a),C(b)) \geq 1$ or $N(C(b),C(a)) \geq 2$,

(b) if a and b are prefix symbols and a and b are siblings, then $N(C(a),C(b)) \geq 1$ or $N(C(b),C(a)) \geq 1$.

Proof. Take any two words x, y of C and write $x = ax'$, $y = by'$ where a, b are prefix symbols and x' is in $C(a)$, y' is in $C(b)$. Now C is 1-asymmetric if and only if $D_a(x,y) \geq 2$ for all such x and y. We will examine this condition for each of the four cases possible for a and b.

(1). If a and b are equal, then for any x and y we have $D_a(x,y) \geq 2$ iff $D_a(x',y') \geq 2$ iff $D_a(C(a)) \geq 2$.

(2). If a and b are siblings, then $N(a,b) = N(b,a) = 1$. Thus $D_a(x,y) \geq 2$ iff $N(x,y) \geq 2$ or

$N(y,x) \geq 2$ iff $N(x',y') \geq 1$ or $N(y',x') \geq 1$ iff $N(C(a),C(b)) \geq 1$ or $N(C(b),C(a)) \geq 1$.

(3). If one of a or b is a parent of the other, assume without loss of generality that a is a parent of b. Then $N(a,b) = 1$ while $N(b,a) = 0$. Thus $D_a(x,y) \geq 2$ iff $N(x',y') \geq 1$ or $N(y',x') \geq 2$ iff $N(C(a),C(b)) \geq 1$ or $N(C(b),C(a)) \geq 2$.

(4). If a and b are not related, then $D_a(a,b) \geq 2$ already so that $D_a(x,y) \geq 2$ automatically. Thus the conditions we presented are both necessary and sufficient.

III. The Constantin-Rao Codes

A completely different approach to 1-asymmetric codes was taken by Constantin and Rao [3] and elaborated by McEliece and Rodemach [5]. This approach is to consider a certain function from a set of binary vectors to an Abelian group and define a code as a preimage of a single group element. Asymmetric errors produce vectors which give the wrong group element under the function; the correction is done by noting which group element is missing and is needed to produce the correct element. Thus this method produces a code which is easily understood and applied. We consider two special results for these codes. First, we relate Constantin-Rao codes to our earlier studies of systematic codes by showing that they are not, in fact, systematic. We then determine necessary and sufficient conditions for these codes to be closed under complements, and relate this result to other results to be given in section IV.

Let V be the set of all binary vectors of length N, and denote by $v[i]$ the i-th element of a vector v of V. Let G be an Abelian group of order $N + 1$ and denote its elements by $g_0, g_1, g_2, \ldots, g_N$ where g_0 is the group identity and the other group elements are arbitrary. Denote by $o(g)$ the order of an element g of G. Define a function T from V to G by:

$$3.1 \qquad T(v) = \sum_{i=1,N} v[i]\, g_i$$

where $v[i]\, g_i = g_i$ if $v[i] = 1$ and $v[i]\, g_i = g_0$ if $v[i] = 0$. For any g in G, then, define $C(g) = \{ v \text{ in } V \text{ such that } T(v) = g \}$. In particular, define $C_0 = C(g_0)$. We have the following results.

3.2. Proposition ([3], Lemma 7). For any g in G, the code $C(g)$ is a 1-asymmetric code.

3.3. Proposition ([3], Theorem 9). The order of C_0 is at least as large as the order of $C(g)$ for any g in G. In particular, the order of C_0 is at least as large as $2^N/(N+1)$.

Since the code C_0 is thus the largest code which can be constructed with this process, we will refer to it as the Constantin-Rao code for a given word length and group. Note, of course, that there can be many Abelian groups with the same order, and so there can be many Constantin-Rao codes with the same word length.

The following theorem determines completely the size of the Constantin-Rao code for a given group.

3.4. Theorem ([5], Theorem 1). For a given word length N and Abelian group G, the size of the Constantin-Rao code is given by

$$\frac{1}{N+1} \sum 2^{(N/o(h))}$$

where the sum is taken over all group elements h of odd order.

Another type of code is the AN-code, where an information symbol N is represented by multiplying it by a constant A and storing the result when the product is taken modulo another constant M. The AN-codes are known not to be systematic; see for example [9], Chapter 3. Further, AN-codes are known to be related to Constantin-Rao codes; see for example [10]. This led us to investigate whether Constantin-Rao codes are systematic, with the following result.

3.5. Proposition. If C is a Constantin-Rao code of length N, where N is not one less than an integral power of 2, then C cannot be a systematic code with fewer than $\log_2(N+1)$ check bits.

Proof. Assume that C is systematic with r check bits and $k = n-r$ information bits, where $r < \log_2(n+1)$.

Assume the code is defined as above by a group G and a function F, and recall that C = { v in V such that $F(v) = g_0$ }. Then for any v in C, we have

$$g_0 = \sum_{\text{info}} v[i]\, g_i + \sum_{\text{check}} v[i]\, g_i .$$

Thus the check bits must generate the group inverses of the elements of G which are generated by the information bits. Now the check bits can generate only 2^r elements, which is fewer than the number of elements in G. However, the information bits generate all of G. For take an arbitrary but fixed element h of G. Then for any g in G, there is a g' in G such that $g + g' = h$. Since $r < \log_2 (n+1) < n/2$, we must have $k > n/2$. Hence there are two information bits with indices i and j such that $g_i + g_j = h$, so that h is generated by information bits alone. This contradiction establishes the result.

The stipulation that N cannot be one less than a power of 2 is in fact necessary, because a Constantin-Rao construction can be used to build a full Hamming code which is, of course, systematic.

As we developed some examples of the Constantin-Rao codes, we noted that sometimes they had the property that any vector v in the code also had its bitwise complement v' in the code. We call this property _closure under complements_. Such closure properties are of some interest, at least in the algebraic sense, and we have the following characterization of this property for Constantin-Rao codes.

3.6. <u>Theorem</u>. Let C be the Constantin-Rao code of length N defined by the Abelian group G of order N+1. Then C is closed under complements if and only if the sum of the elements in G is the identity.

<u>Proof</u>. Let the code C have n code words, and let the order of G be n + 1. For any code word v in C, let v' be the bitwise complement of v. Then v'[i] = 1-v[i] for each index i. We thus have

$$v' \text{ is in } C \text{ iff } T(v') = 0 \text{ iff } \sum_{n} (1-v[i])\, g_i = g_0$$

$$\text{iff } \sum_{n} g_i - \sum_{n} v[i]\, g_i = g_0 \text{ iff } \sum_{n} g_i = g_0$$

since $T(v) = g_0$.

3.7. <u>Corollary</u>. If C is a Constantin-Rao code of even length, then C is closed under complements.

<u>Proof</u>. Since C is of even length, the group G used in the code construction must be of odd order. Thus every non-identity element of G must be of odd order. Define subsets S and T of G as follows: for any g in G, g is in S if and only if -g is in T. Then the intersection of S and T is simply $\{g_0\}$, while the sum of the elements in S is clearly the inverse of the sum of the elements in T. However, the sum of S and T is also the sum of G, thus showing that the sum of the group elements is g_0.

Recall that an _elementary Abelian 2-group_ is an Abelian group in which each element has order one or two. Such groups are a direct sum of a number of copies of the group of order two.

3.8. _Corollary_. If C is a Constantin-Rao code of length 2^k-1 defined by an elementary Abelian 2-group, for $k > 1$, then C is closed under complements.

Proof. A straightforward computation shows that in any single component of such a group, there are $2^{(k-1)}$ 1's. Since k is greater than 1, the sum in each component must be 0. Thus the sum of the elements of the group is g .

3.9. _Corollary_. If C is a Constantin-Rao code defined by a group which is the additive group of a Galois field GF(q), $q > 2$, then C is closed under complements.

Proof. Each GF(q) has an additive group which is either of odd order or is an elementary Abelian 2-group. The result thus follows from Corollaries 3.7 and 3.8.

This corollary leads us to ask, in the next chapter, whether codes defined by similar techniques for the Galois fields are closed under complements.

Finally, we give a class of Constantin-Rao codes which are not closed under complements.

3.10. Corollary. If C is a Constantin-Rao code of odd length defined by a group which is cyclic of even order, then C is not closed under complements.

Proof. For such a group, construct the sets S and T as in the proof of Corollary 3.7. Then the (unique) element of order 2 in the group lies in both S and T, and it is quickly seen that the sum of the group elements is this element of order 2 instead of g_0.

IV. A Code Correcting Multiple Asymmetric Errors

In this chapter we define a code C which corrects multiple asymmetric errors. This code is built by a method which extends those of Constantin and Rao and of Graham and Sloane [3]. We verify that this code is t-asymmetric for suitable t. We then show that such codes do not lend themselves to the clean characterization of the Constantin-Rao codes, but they do sometimes provide more code words for a given code length than standard symmetric codes.

Let q be a prime power integer and $GF(q)$ be the Galois field with q elements. Write $GF(q) = \{ g_0, g_1, \ldots, g_p \}$ with $p = q-1$, and let V be the set of all binary vectors of length $q-1$ with a member v of V written as $< v_1, v_2, \ldots, v_p >$. Note that we allow vectors of any weight, while Graham and Sloane considered only constant-weight binary vectors. For any k with $0 < k < q$ define the function T_k from V to $GF(q)$ by

4.1 $$T_k(v) = \sum g_{i(1)} * g_{i(2)} * \ldots * g_{i(k)}$$

where the sum is taken over all indices $i(1),\ldots,i(k)$ with $i(1)<i(2)<\ldots<i(k)$ and $v_{i(1)}=\ldots=v_{i(k)}=1$. Note

that T_1 is the Constantin-Rao function. Denote by $GF^m(q)$ the direct sum or product of m copies of $GF(q)$, and define T from V to $GF^m(q)$ to be the function whose coordinate functions are $\{ T_k \}$ for k ranging from 1 to m. For any $w = < w_1, w_2, \ldots, w_m >$ in $GF^m(q)$ define $C_w = \{ v \text{ in } V \text{ with } T(v) = w \}$. We shall show that each of the C_w is an m-asymmetric code and shall study the properties of these codes.

Before we can establish the fact that such codes are m-asymmetric, we must have a technical result. For a, b in V, we define three subsets Q, R, and S of $\{ 1..q-1 \}$ as follows: $Q = \{ i \text{ with } a_i = b_i \}$, $R = \{ i \text{ with } a_i = 1 \text{ and } b_i = 0 \}$, and $S = \{ i \text{ with } a_i = 0 \text{ and } b_i = 1 \}$. Denote by a' and b' those elements of V which have 1-bits precisely for the indices in R and S, respectively, and let c be the element of V which agrees with a and b on the bits with indices in Q but is 0 for all other indices.

4.2. Lemma. If a and b are elements of V such that $T(a) = T(b)$, then for any positive integer $k \leq m$ we have $T_k(a') = T_k(b')$.

Proof. Assume that $T(a) = T(b)$.. We argue by induction on k. If $k = 1$, then $T_1(a) = T_1(a') + T_1(c)$ while $T_1(b) = T_1(b') + T_1(c)$. Thus $T_1(a') = T_1(b')$ easily.

Now assume that for every $j < k$ we have

$T_j(a') = T_j(b')$. Write $T_k(a) = \sum g_{i(1)}*g_{i(2)}*..*g_{i(k)}$ and note that in any such product, certain indices are in Q and others are in R. Now GF(q) is commutative so we may reorder and rewrite this product as $T_k(a) = \sum T_j(a')*T_{(k-j)}(c)$ with the sum taken over all j from 0 to k, where $T_0(v) = 1$ by the usual logic of empty products being 1. Similarly, we may write $T_k(b) = \sum T_j(b')*T_{(k-j)}(c)$. Since our inductive hypothesis gives us $T_j(a') = T_j(b')$ for all $j < k$, it follows easily that $T_k(a') = T_k(b')$.

Note now that R contains N(a,b) elements and S contains N(b,a) elements. Thus to prove that the C_w are m-asymmetric codes, we will prove that either R or S must contain at least m+1 elements.

4.3. Theorem. The code C_w defined by T from V to $GF^m(q)$ is an m-asymmetric code.

Proof. Take any a, b in C_w with a <> b, and define R, S, a', and b' as above. Assume both R and S have fewer than m+1 elements and have unequal sizes; without loss of generality assume R has n elements while S has fewer. Then $T_n(a') <> 0$ while $T_n(b') = 0$, which contradicts Lemma 4.2. Hence if R and S have fewer than m+1 elements, they must have equal sizes.

Now assume each of R and S contains n elements, for $n \leq m$. Let $R' = \{ r_1, r_2, \ldots, r_n \}$ and $S' = \{ s_1, s_2, \ldots, s_n \}$ be the subsets of $GF(q)$ which are the images of the weight-1 vectors whose coordinates are in R and S, respectively. Denote by c_k the value of the k-th elementary symmetric function on R'. Since $T_k(a') = T_k(b')$ for all $k \leq n$, c_k is also the value of the k-th elementary symmetric function on S'. Defining $c_0 = 1$, we then see that all of $r_1, \ldots, r_n, s_1, \ldots, s_n$ are roots of the polynomial

$$P(X) = \sum_{i=0}^{n} (-1)^{i} * c_{i} * x^{(n-i)} .$$

Thus $P(X)$ is a polynomial of degree n having $2n$ roots, which is impossible. Hence one of R or S must contain at least $m+1$ elements, so that $D_a(C_w) \geq m+1$. Thus C_w is a m-asymmetric code.

Since C_w is an m-asymmetric code for any w in $GF^m(q)$, we may choose w so that C_w is maximal. A simple counting argument establishes the following corollary. Note that since the BCH codes correcting m errors contain exactly $2^{(q-1)}/q^m$ elements, this corollary shows that our codes have information rates at least as high as BCH codes.

4.4. <u>Corollary</u>. There exist m-asymmetric codes of length $q-1$ and size at least $2^{(q-1)}/q^m$ for any prime power q and any $m < (q-1)/\log_2(q)$. These codes have information rates at least $r = 1-m*\log_2(q)/(q-1)$.

In fact, the codes C defined here are m-asymmetric and need not be (m+1)-asymmetric. Let V be the set of binary vectors of length 6 and consider the code C = { v in V such that T(v) = 0 } for T mapping V to $GF^2(7)$. This is the largest such 2-asymmetric code, $D_a(C) = 3$ exactly, and C contains 4 elements. This is slightly better than the lower bound of 2 given in Corollary 4.4. Moreover, the best code of length 6 correcting double symmetric errors contains only two elements.

Now the m-asymmetric codes we have defined here extend both the concept and the construction of the Constantin-Rao 1-asymmetric codes. Thus it is appropriate to ask how much of the Constantin-Rao theory extends to these m-asymmetric codes. Specifically, we ask three questions.

1. Does the m-asymmetric code have a better information rate than known codes correcting m symmetric errors?

2. Is the maximal m-asymmetric code defined here the preimage of the zero vector in $GF^m(q)$?

3. Since the Constantin-Rao codes are closed under complements for all additive groups of Galois fields, are these m-asymmetric codes closed under complements?

These questions are all examined by constructing examples of such codes. Our examples are limited somewhat by the 16-bit architecture we have available and by the computer time available to us.

We constructed a number of the m-asymmetric codes of length n for small m and n and measured their properties; some programs for this are contained in the Appendix. Specifically, for the values we examined we first determined the largest code and the particular elements of $GF^m(q)$ giving that code. The following table answers the question of information rates by giving the size of the largest codes found. The information on symmetric codes is from [7], page 124.

Code Sizes for Given Code Lengths

length	double error correcting		
	symmetric	asymmetric	Corollary 4.4
6	2	4	2
7	2	2	2
8	4	6	4
10	8	10	9
12	16	29	25
16	256	231	227
18	512	748	727
22	8192	7946	7929

Table 1

	triple error correcting		
6	n/a	2	1
7	2	2	1
8	2	4	1
10	2	4	1
12	4	6	2
16	32	30	14
18	128	60	39
22	2048	422	345

Table 2

Note that our construction provides codes which sometimes, but not always, have more code words and hence a better information rate than the best symmetric codes for these lengths. This is particularly so for 2-asymmetric codes.

The second question, concerning the elements of $GF^m(q)$ which give the maximal m-asymmetric codes, is more difficult. The Constantin-Rao result we quoted as Theorem 3.3 uses fairly straightforward group properties, though the McEliece-Rodemach determination of the size of the code which we quoted as Theorem 3.4 uses the orthogonality relation among group characters. However, to answer our question fully would require a knowledge of precisely which values are taken most often by each of the symmetric polynomials of degree no larger than m over GF(q); this is a combinatoric question on finite fields whose answer does not appear to be known. Our experimental results indicate that there may be no pattern to this answer, but show quite clearly that the preimage of the zero vector in $GF^m(q)$ is not the maximal code in general. We summarize our results in the following table. Note that the element listed may not be unique; if it is not, we simply list the element which occurs first lexicographically.

Element with the Maximal Preimage

size	2-asymmetric	3-asymmetric
6	<0,0>	<2,5,3>
7	<0,0>	<0,0,0>
8	<0,1>	<1,2,0>
10	<1,3>	<0,5,4>
12	<1,7>	<0,8,9>
16	<0,3>	<0,7,11>
18	<0,0>	<18,11,5>
22	<1,11>	<19,15,16>

Table 3

Finally, we look at the question of closure under complements. More specifically, we ask about closure for two such codes: the maximal code and the code which is the preimage of the zero vector in $GF^m(q)$. Our results are given in the table below.

Closure Under Complements

	2-asymmetric		3-asymmetric	
size	maximal	<0,0>	maximal	<0,0,0>
6	yes	yes	no	n/a
7	yes	yes	yes	yes
8	no	yes	no	n/a
10	no	yes	no	n/a
12	no	yes	no	no
16	no	yes	no	no
18	yes	yes	no	no
22	no	yes	no	no

Table 4

Note that the only cases in which the maximal 2-asymmetric code is closed under complements are those when it is the <0,0> code. These results indicate that the <0,0> code may be closed under complements, and we so conjecture; however, the other codes are not closed under complements.

V. BIBLIOGRAPHY

[1] Bose, B. Theory and Design of Unidirectional Error Codes, Ph.D. Thesis, Southern Methodist University, 1980.

[2] Constantin, Serban N. and T. R. N. Rao, On the Theory of Binary Asymmetric Error Correcting Codes, Information and Control, 40(1979), 20-36.

[3] Graham, R. L. and N. J. A. Sloane, Lower Bounds for Constant Weight Codes, IEEE Transactions on Information Theory, IT-26(1980), 37-43.

[4] Kløve, Torliev, A Lower Bound for A(n,4,w), IEEE Transactions on Information Theory, IT-27(1981), 257-258.

[5] McEliece, Robert J. and Eugene R. Rodemach, The Constantin-Rao Construction for Binary Asymmetric Error-Correcting Codes, Information and Control, 44(1980), 187-196.

[6] McEliece, Robert J., Comment on "A Class of Codes for Asymmetric Channels and a Problem from the Additive Theory of Numbers", _IEEE Transactions on Information Theory_, IT-19(1973), 137.

[7] Peterson, W. Wesley and E. J. Weldon, Jr., _Error Correcting Codes_, 2nd edition, MIT Press, Cambridge, MA, 1972.

[8] Rao, T. R. N. and A. S. Chawla, Asymmetric Error Codes for Some LSI Semiconductor Memories, _The Annual Southeastern Symposium on System Theory_ (1975), 170-171.

[9] Rao, T. R. N., _Error Coding for Arithmetic Processors_, Academic Press, New York, 1974.

[10] Shiozaki, Akira, Single Asymmetric Error Correcting Cyclic AN Codes, to appear.

[11] Varshamov, R. R., A Class of Codes for Asymmetric Channels and a Problem from the Additive Theory of Numbers, _IEEE Transactions on Information Theory_, IT-19(1973), 92-95.

VI. APPENDIX

In this Appendix we present two of the experimental programs we used in examining systematic asymmetric codes and m-asymmetric codes. The programs were originally written in C for the Oregon State University Computer Science Department's PDP 11/40 running UNIX or in FORTRAN for the CDC CYBER system. However, the programs presented are in the form used on the HP-3000 at Birmingham-Southern College and are in Pascal or FORTRAN.

```
program sys85(input,output);
{
   This is a translation into Pascal of the program
   all85.c which was written at Oregon State during
   the summer of 1981.  This translation is used in
   several experimental studies on systematic codes.

   R.  S.  Cunningham, September 21, 1981.

   Bit-shift capabilities in Pascal would improve
   this program if they were available.
}

  const     NVEC   = 32;    { number of code words }
            NINFO  =  5;    { number of information bits }
            CKVAL  =  8;    { number of check symbols }
            NCHECK =  3;    { number of check bits }

  var   i, j : integer;
        info : array[1..NVEC,1..NINFO] of integer;
        check: array[1..NVEC,1..NCHECK] of integer;

  function TTT(j : integer) : integer;
  {
     function to compute the value of 2**j.
  }
    var  k, temp  : integer;

    begin { TTT }
      temp := 1;
      for k := 1 to j do temp := temp * 2;
      TTT := temp
    end;

  function checkem(i, j : integer) : boolean;
  {
     Function which computes the asymmetric distance
     between the i-th and j-th code words in a code
     being considered and returns either true (the
     asymmetric distance is at least 2) or false (the
     asymmetric distance is less than 2).
  }
    var   a, b, k  : integer;
          temp_check : boolean;

    begin { checkem }
      a := 0;  b := 0;   { the one-directional distances
}
```

```
      for k := 1 to NINFO do begin
        if info[i,k] < info[j,k] then b := b+1;
        if info[i,k] > info[j,k] then a := a+1
      end;
      for k := 1 to NCHECK do begin
        if check[i,k] < check[j,k] then b := b+1;
        if check[i,k] > check[j,k] then a := a+1
      end;
      checkem := (a > 1) or (b > 1);
    end;

  procedure build_row(i : integer);
  {
     Recursive procedure which prints the check values of
     a code if the code has been checked out as valid
past
     the end of the code array.  This procedure uses a
     backtracking method to reach the end of this array
     if possible.

     This procedure can be modified to check other
     asymmetric distances for further experiments.
  }
    var  j, k, sum, check_sum    : integer;
         flag                    : boolean;

    begin { build_row }
      if i > NVEC then begin       { successful code }
        for k := 1 to NVEC do begin
          sum := 0;
          for j := 1 to NCHECK do
            sum := sum + check[k,j] * TTT(NCHECK-j);
          write(sum:3)
        end;
        writeln; writeln;
      end
      else begin       { code not yet finished }
        check_sum := 0;
        while check_sum < CKVAL do begin
          for j := 1 to NCHECK do
            check[i,j] :=
       (check_sum div TTT(NCHECK-j)) mod 2;
          check_sum := check_sum + 1;
          flag := true;
          j := 1;
          while flag and (j<i) do begin
            flag := checkem(i,j);
            j := j + 1
          end;
          if flag then build_row(i+1)
        end
      end
    end; { build_row }
```

```
begin { main program }
  writeln('(',NCHECK+NINFO:2,',',NINFO:2,') codes:');
  writeln('check symbols are');
  writeln;
  for i := 0 to NVEC-1 do
    write(i:3);    { print information symbols }
  writeln;
  for i := 0 to NVEC-1 do
    write('___');
  writeln; writeln;
  for i := 1 to NVEC do
    for j := 1 to NINFO do
      info[i,j] := ((i-1) div TTT(NINFO-j)) mod 2;
  build_row(1)
end.
```

```
$INTEGER*4
      PROGRAM GF17
C****************************************************************
C
C     PROGRAM TO TEST THE 3-ASYMMETRIC-ERROR-CORRECTING
C     CODE DEFINED BY V(16) --> GF(17,3) FOR ITS SIZE
C     AND OTHER PROPERTIES.
C     ORIGINALLY WRITTEN JULY, 1981 FOR THE OSU CYBER
C     SYSTEM; REWRITTEN SEPTEMBER 1981 FOR THE B-SC
C     HP3000 FOR GF(11) AND SUBSEQUENTLY MODIFIED FOR
C     SEVERAL OTHER CASES.
C
C****************************************************************
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION CODE(16),RESULTS(17,17,17),ROWSUM(17,17)
      DATA RESULTS/4913*0/
      WRITE(6,5)
    5 FORMAT(1H1,8X,"EXPERIMENTAL RESULTS :",
     &  " V(16) 2- AND 3-ASYMMETRIC CODES.",/)
      N=16
      NP1=N+1
      DO 300 I = 0, 2**N-1
        VALUE=I
        DO 10 J=1,N
          CODE(J)=MOD(VALUE,2)
          VALUE=VALUE/2
 10     CONTINUE
        SUM1=0
        SUM2=0
        SUM3=0
        DO 100 J=1,N
          IF (CODE(J).EQ.0) GOTO 100
          SUM1=MOD(SUM1+J,NP1)
          IF (J.EQ.N) GOTO 100
          DO 50 K=J+1,N
            IF (CODE(K).EQ.0) GOTO 50
            TEMP=MOD(J*K,NP1)
            SUM2=MOD(SUM2+TEMP,NP1)
            IF (K.EQ.N) GOTO 50
            DO 20 L=K+1,N
              IF (CODE(L).EQ.0) GOTO 20
              TEMP=MOD(TEMP*L,NP1)
              SUM3=MOD(SUM3+TEMP,NP1)
 20         CONTINUE
 50       CONTINUE
100     CONTINUE
```

```
          RESULTS(SUM1+1,SUM2+1,SUM3+1)=
        & RESULTS(SUM1+1,SUM2+1,SUM3+1)+1
 300     CONTINUE
         WRITE(6,325)
 325     FORMAT(/," VALUE COUNTS FOR CODE DETERMINATION"/
        & " ROWS ARE CONSTANT VALUES OF SECOND TERMS"/)
         DO 335 I=1,NP1
           DO 333 J=1,NP1
             ROWSUM(I,J)=0
             DO 331 K=1,NP1
               ROWSUM(I,J)=ROWSUM(I,J)+RESULTS(I,J,K)
 331         CONTINUE
 333       CONTINUE
 335     CONTINUE
         DO 500 I=1,NP1
           WRITE(6,350) I-1
 350       FORMAT(/," COUNTS FOR PLANE",I3,40X,"ROW SUM")
           WRITE(6,400) ((RESULTS(I,J,K),K=1,NP1),
        &      ROWSUM(I,J),J=1,NP1)
 400       FORMAT(17(3X,17I3,3X,I3/))
 500     CONTINUE
         MAX=0
         DO 600 I=1,NP1
           DO 570 J=1,NP1
             DO 540 K=1,NP1
               IF (RESULTS(I,J,K).LE.MAX) GOTO 540
               MAX=RESULTS(I,J,K)
               SUM1=I
               SUM2=J
               SUM3=K
 540         CONTINUE
 570       CONTINUE
 600     CONTINUE
         WRITE(6,700) MAX, SUM1-1,SUM2-1,SUM3-1
 700     FORMAT(/" LARGEST CODE HAS ",I5," WORDS;",
        &        " COORDINATES ARE: ",3I4/1H1)
         STOP
         END
```