Chapter 14

Some error-correcting codes and their applications

J. D. Key[1]

14.1 Introduction

In this chapter we describe three types of error-correcting linear codes that have been used in major applications, *viz.* photographs from spacecraft (first order Reed-Muller codes), compact discs (Reed-Solomon codes), and computer memories (extended binary Hamming codes).

Error-correcting codes were first developed in the 1940s following a theorem of Claude Shannon [14] that showed that almost error-free communication could be obtained over a noisy channel. The message to be communicated is first "encoded", i.e. turned into a codeword, by adding "redundancy". The codeword is then sent through the channel and the received message is "decoded" by the receiver into a message resembling, as closely as possible, the original message. The degree of resemblance will depend on how good the code is in relation to the channel.

Such codes have been used to great effect in some important applications, and we will describe here the codes that are used in three of these applications, showing how they can be constructed and how they can be used:

- **Computer memories** [11]: the codes used are extended binary Hamming codes, the latter being perfect single-error-correcting;
- **Photographs from spacecraft**: the codes initially used were first-order Reed-Muller codes, which can be constructed as the orthogonal extended Hamming codes; later the binary extended Golay code was used;
- **Compact discs** [7]: the codes used are Reed-Solomon codes, constructed using certain finite fields of large prime-power order.

[1]Chapter 14 of "Applied Mathematical Modeling: A Multidisciplinary Approach", D. R. Shier and K. T. Wallenius (Eds.), Chapman & Hall/CRC Press, Boca Raton, FL, 1999

After an introductory section on the necessary background to coding theory, including some of the effective encoding and decoding methods, we will describe how the codes can be used in each of these applications, and give a simple description how each of these classes of codes can be constructed. We will not include details of the implementation of the codes, nor of the mathematical background to the theory; the reader is encouraged to consult the papers and books in the bibliography for this. Those readers who are familiar with the elementary concepts in coding theory should pass immediately on to the applications, and refer back to Section 14.2 when necessary. The final section contains some simple exercises and some projects for further study.

14.2 Background coding theory

More detailed accounts of error-correcting codes can be found in: Hill [6], Pless [13], MacWilliams and Sloane [10], van Lint [9], and Assmus and Key [1, Chapter 2]. See also Peterson [12] for an early article written from the engineers' point of view. Proofs of all the results quoted here can be found in any of these texts; our summary here follows [1].

The usual pictorial representation of the use of error-correcting codes to send messages over noisy channels is shown in the schematic diagram Figure 14.1.

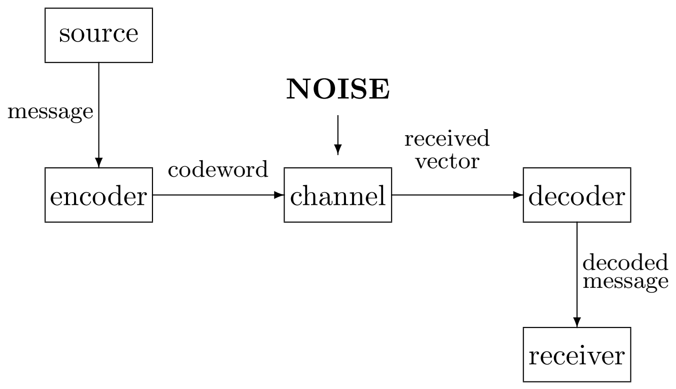

Figure 14.1: A noisy communications channel

Here a message is first given by the *source* to the *encoder* that turns the message into a *codeword*, i.e. a string of letters from some alphabet, chosen according to the code used. The encoded message is then sent through the channel, where it may be subjected to *noise* and hence altered. When this message arrives at the *decoder* belonging to the receiver, it is equated with the most likely codeword, i.e. the one (should that exist) that, in a probabilistic sense depending on the channel, was probably sent, and finally this "most likely" codeword is decoded and the message is passed on to the receiver.

Example 14.1 Suppose we use an alphabet of just two symbols, 0 and 1, and we have only two messages, for example "no" corresponding to 0, and "yes" corresponding to 1. We wish to send the message "no", and we add redundancy by simply repeating the message five times. Thus we encode the message as the codeword (00000). The channel might interfere with the message and could change it to, say, (10100). The decoder assesses the message and decides that of the two possible codewords, i.e. (00000) and (11111), the former is the more likely, and hence the message is decoded, correctly, as "no".

We have made several assumptions here: for example we have assumed that the probability of an error at any position in the word is less than $\frac{1}{2}$, that each codeword is equally likely to be sent, and that the receiver is aware of the code used.

Definition 14.1 *Let F be a finite set, or* **alphabet**, *of q elements. A* **q-ary code** *C is a set of finite sequences of symbols of F, called* **codewords** *and written $x_1x_2 \ldots x_n$, or $(x_1, x_2, \ldots, x_n)$, where $x_i \in F$ for $i = 1, \ldots, n$. If all the sequences have the same length n, then C is a* **block code** *of* **block length** *n.*

The code used in the example above is a block code, called the **repetition code** of length 5: it can be generalized to length n and to any alphabet of size q, and hence will have q codewords of the form $xx \cdots x$, where $x \in F$.

Given an alphabet F, it will be convenient, and also consistent with terminology for cartesian products of sets and for vector spaces when F is a field, to denote the set of all sequences of length n of elements of F by F^n and to call these sequences *vectors*, referring to the member of F in the i^{th} position as the coordinate at i. We use either notation, $x_1x_2 \cdots x_n$ or $(x_1, x_2, \ldots, x_n)$, for the vectors. A code over F of block length n is thus any subset of F^n.

The formal process in the reasoning of the simple example given above uses the concept of the *distance* between codewords.

Definition 14.2 *Let $v = (v_1, v_2, \ldots, v_n)$ and $w = (w_1, w_2, \ldots, w_n)$ be two vectors in F^n. The* **Hamming distance**, *$d(v, w)$, between v and w is the number of coordinate places in which they differ:*

$$d(v, w) = |\{i | v_i \neq w_i\}|.$$

We will usually refer to the Hamming distance as simply the **distance** between two vectors. It is simple to prove that the Hamming distance defines a metric on F^n, i.e.

(1) $d(v, w) = 0$ if and only if $v = w$;

(2) $d(v, w) = d(w, v)$ for all $v, w \in F^n$;

(3) $d(u, w) \leq d(u, v) + d(v, w)$ for all $u, v, w \in F^n$.

Nearest-neighbor decoding picks the codeword v' nearest (in terms of Hamming distance) to the received vector, should such a vector be uniquely determined. This method maximizes the decoder's likelihood of correcting errors — provided that each symbol has the same probability (less than $\frac{1}{2}$) of being received in error and each symbol of the alphabet is equally likely to occur. A channel with these two properties is called a **symmetric q-ary channel**.

Definition 14.3 *The* **minimum distance** $d(C)$ *of a code C is the smallest of the distances between distinct codewords; i.e.*

$$d(C) = \min\{d(v, w) | v, w \in C, v \neq w\}.$$

For the repetition code of length 5 given in Example 14.1 this distance is 5.

The following simple result is very easily proved and shows the vital importance of this concept for codes used in symmetric channels.

Theorem 14.1 *If $d(C) = d$ then C can detect up to $d - 1$ errors or correct up to $\lfloor (d-1)/2 \rfloor$ errors.*

(Here $\lfloor n \rfloor$ denotes the floor function of n.) Thus in Example 14.1 up to four errors can be detected or up to two errors can be corrected.

If C is a code of block length n having M codewords and minimum distance d, then we say that C is an **(n, M, d) q-ary code**, where $|F| = q$. We will refer to n as the **length** of the code rather than the block length. Thus the code in Example 14.1 is a $(5, 2, 5)$ 2-ary (binary) code.

From the above discussion, we see that for a good (n, M, d) code C, i.e. one that detects or corrects many errors, we need d to be large. However we also prefer n to be small (for fast transmission) and M to be large (for a large number of messages). These are clearly conflicting aims, since for a q-ary code, $M \leq q^n$. In fact there are many bounds connecting these three parameters, one of the simplest of which is the **Singleton bound** (see, for example, [1, Theorem 2.1.2]):

$$M \leq q^{n-d+1}. \tag{14.1}$$

Another bound, usually better than the Singleton bound, is the **sphere-packing bound**.

Definition 14.4 *Let F be any alphabet and suppose $u \in F^n$. For any integer $r \geq 0$, the* **sphere of radius r with center u** *is the set of vectors $S_r(u) = \{v | v \in F^n, d(u, v) \leq r\}$.*

Let C be an (n, M, d) code. Then the spheres of radius $\rho = \lfloor (d-1)/2 \rfloor$ with center in C do not overlap, i.e. they form M pairwise disjoint subsets of F^n. The integer ρ is called the **packing radius** of C. Hence we have the **sphere-packing bound**: if C is an (n, M, d) q-ary code of packing radius ρ, then

$$M\left(1 + (q-1)n + (q-1)^2\binom{n}{2} + \cdots + (q-1)^\rho \binom{n}{\rho}\right) \leq q^n. \tag{14.2}$$

The **covering radius** of a code is defined to be the smallest integer R such that spheres of radius R with their centers at the codewords cover all the words of F^n. If the covering radius R is equal to the packing radius ρ, the code is called a **perfect ρ-error-correcting code**. Thus perfect codes are those for which equality holds in (14.2)

The binary (i.e. $|F| = 2$) repetition codes of odd length n are *trivial* perfect $(n-1)/2$-error-correcting codes; the infinite class of binary perfect 1-error-correcting codes with $n = 2^m - 1$ and hence $M = 2^{n-m}$ was discovered by Hamming [4] and generalized to the q-ary case by Golay.

A code C over the finite field $F = \mathbf{F}_q$ of prime-power order q, of length n is **linear** if C is a subspace of $V = F^n$. If $\dim(C) = k$ and $d(C) = d$, then we write $[n, k, d]$ or $[n, k, d]_q$for the q-ary code C; if the minimum distance is not specified we simply write $[n, k]$. The **information rate** is k/n and the **redundancy** is $n - k$.

Thus a q-ary linear code is any subspace of a finite-dimensional vector space over a finite field $\mathbf{F}_q$, *but with reference to a particular basis.* The standard basis for F^n, as the space of n-tuples, has a natural ordering through the numbers 1 to n, and this coincides with the spatial layout of a codeword as a sequence of alphabet letters sent over a channel. To avoid ordering the basis we may take $V = F^X$, the set of functions from X to F, where X is any set of size n. Then a linear code is any subspace of V.

For any vector $v = (v_1, v_2, \ldots, v_n) \in V = F^n$, let $S = \{i \,|\, v_i \neq 0\}$; then S is called the **support** of v, written $\mathrm{Supp}(v)$, and the **weight** of v, $\mathrm{wt}(v)$, is $|S|$. The **minimum weight** of a code is the minimum of the weights of the non-zero codewords, and for linear codes is easily seen to be equal to $d(C)$.

For linear $[n, k, d]$ q-ary codes the Singleton bound and the Sphere-packing bound become the following:

Singleton bound: $d \leq n - k + 1$;

sphere-packing bound: $\sum_{i=0}^{\rho} (q-1)^i \binom{n}{i} \leq q^{n-k}$.

A code for which equality holds in the Singleton bound is called an MDS (maximum distance separable) code. The Reed-Solomon codes (see Section 14.5.2) are MDS codes.

Definition 14.5 *Two linear codes in F^n are* **equivalent** *if each can be obtained from the other by permuting the coordinate positions in F^n and multiplying each coordinate by a non-zero field element. The codes will be said to be* **isomorphic** *if a permutation of the coordinate positions suffices to take one to the other.*

In terms of the distinguished basis that is present when discussing codes, code equivalence is obtained by reordering the basis and multiplying each of the basis elements by a non-zero scalar. Thus the codes C and C' are equivalent if there is a linear transformation of the ambient space F^n, given by a *monomial* matrix (one non-zero entry in each row and column) in the standard basis, that carries C onto C'. When the codes are isomorphic, a *permutation* matrix can be found with this property. When $q = 2$, the two concepts are identical. Clearly equivalent linear codes must have the same parameters $[n, k, d]$.

If C is a q-ary $[n, k]$ code, a **generator matrix** for C is a $k \times n$ array obtained from any k linearly independent vectors of C.

Via elementary row operations, a generator matrix G for C can be brought into reduced row echelon form and still generate C, and then, by permuting columns, it can be brought into a **standard form**

$$G' = [I_k | A] \tag{14.3}$$

where this is now a generator matrix for an equivalent (in fact, isomorphic) code. Here A is a $k \times (n - k)$ matrix over F.

Now we come to another important way of describing a linear code, *viz.* through its *orthogonal* or *dual*. For this we need an *inner product* defined on our space; it is the standard inner product: for $v, w \in F^n$, $v = (v_1, v_2, \ldots, v_n)$, $w = (w_1, w_2, \ldots, w_n)$, we write the inner product of v and w as (v, w) where

$$(v, w) = \sum_{i=1}^{n} v_i w_i. \tag{14.4}$$

Definition 14.6 *Let C be a q-ary $[n, k]$ code. The* **orthogonal code** *(or* **dual code***) is denoted by $C^{\perp}$ and is given by*

$$C^{\perp} = \{v \in F^n | (v, c) = 0 \text{ for all } c \in C\}.$$

We call C **self-orthogonal** *if $C \subseteq C^{\perp}$ and* **self-dual** *if $C = C^{\perp}$.*

From elementary linear algebra we have

$$\dim(C) + \dim(C^{\perp}) = n \tag{14.5}$$

since $C^{\perp}$ is simply the null space of a generator matrix for C. Taking G to be a generator matrix for C, a generator matrix H for $C^{\perp}$ satisfies $GH^t = 0$, i.e. $c \in C$ if and only if $cH^t = 0$, or, equivalently, $Hc^t = 0$. Any generator matrix H for $C^{\perp}$ is called a **parity-check** or **check** matrix for C. If G is written in the standard form

$$[I_k | A],$$

then

$$H = [-A^t | I_{n-k}] \tag{14.6}$$

is a check matrix for the code with generator matrix G.

In Example 14.1, which is a linear $[5, 1, 5]_2$ code, we have generator matrix

$$G = [1, 1, 1, 1, 1],$$

already in standard form, and thus

$$H = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

as a check matrix.

A generator matrix in standard form simplifies encoding: suppose data consisting of q^k messages are to be encoded by adding redundancy using the code C with generator matrix G. First identify the data with the vectors in F^k, where $F = \mathbf{F}_q$. Then for $u \in F^k$, encode u by forming uG. If $u = (u_1, u_2, \ldots, u_k)$ and G has rows $R_1, R_2, \ldots, R_k$, where each R_i is in F^n, then $uG = \sum_i u_i R_i = (x_1, x_2, \ldots, x_k, x_{k+1}, \ldots, x_n) \in F^n$, which is now encoded. But when G is in standard form, the encoding takes the simpler form

$$u \mapsto (u_1, u_2, \ldots, u_k, x_{k+1}, \ldots, x_n),$$

and here the $u_1, \ldots, u_k$ are the *message* or *information* symbols, and the last $n-k$ entries are the *check* symbols, and represent the redundancy.

In general it is not possible to say anything about the minimum weight of $C^\perp$ knowing only the minimum weight of C but, of course, either a generator matrix or a check matrix gives complete information about both C and $C^\perp$. In particular, a check matrix for C determines the minimum weight of C in a useful way:

Theorem 14.2 *Let H be a check matrix for an $[n, k, d]$ code C. Then every choice of $d-1$ or fewer columns of H forms a linearly independent set. Moreover if every $d-1$ or fewer columns of a check matrix for a code C are linearly independent, then the code has minimum weight at least d.*

Notice that in terms of generator matrices, two codes C and C' with generator matrices G and G' are equivalent if and only if there exist a non-singular matrix M and a monomial matrix N such that $MGN = G'$, with isomorphism if N is a permutation matrix and equality if $N = I_n$, n being the block length of the codes.

Example 14.2 The smallest non-trivial Hamming code (see Section 14.3.2) is a [7,4,3] binary code, which is a perfect single-error-correcting code. It can be given by the generator matrix G in standard form $[I_4|A]$ where

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

Thus a check matrix will be

$$H = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Taking $\{a, b, c, d\}$ as the information symbols, and $\{b', c', d'\}$ as the check symbols, the diagram shown in Figure 14.2 (due to McEliece — see, for example, [11]) can be used to correct a single error, for any vector received, if at most a single error has occurred. The rule is that the sum of the coordinates in any of the three

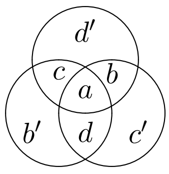

Figure 14.2: The Hamming code $\mathcal{H}_3$

circles must be 0, which constitute the "parity checks" as seen from the matrix H above. Thus, for example, if the vector 1011111 is received, checking the parity in the three circles shows that an error occurred at the information symbol b, so that the error is corrected, yielding 1111111.

A general method of decoding for linear codes is a method, due to Slepian [15], that uses nearest-neighbor decoding and is called **standard-array decoding**. The **error vector** is defined to be $e = w - v$, where v is the codeword sent and w is the received vector. Given the received vector we wish to determine the error vector. We look for that coset of the subgroup C in F^n that contains w and observe that the possible error vectors are just the members of this coset. The strategy is thus to look for a vector e of minimum weight in the coset $w + C$, and decode as $v = w - e$. A vector of minimum weight in a coset is called a **coset leader**; of course it might not be unique, but it will be in the event that its weight is at most ρ, where ρ is the packing radius, and this will always happen when at most ρ errors occurred during transmission. It should be noted that there may be a unique coset leader even when the weight of that leader is greater than ρ and thus a complete analysis of the probability of success of nearest-neighbor decoding will involve analyzing the weight distribution of all the cosets of C; in the engineering literature this is known as "decoding beyond the minimum distance". Use of a parity-check matrix H for C to calculate the **syndrome**, *viz.*

$$\mathrm{synd}(w) = wH^t, \tag{14.7}$$

of the received vector w assists this decoding method, the syndrome being constant over a coset and equal to the zero vector when a codeword has been received.

Definition 14.7 *Let C be a q-ary $[n, k, d]$ code. Define the* **extended** *code $\widehat{C}$ to be the code of length $n+1$ in F^{n+1} of all vectors $\hat{c}$ for $c \in C$ where, if $c = (c_1, \ldots, c_n)$, then*

$$\hat{c} = \left(c_1, \ldots, c_n, -\sum_{i=1}^{n} c_i\right).$$

This is called **adding an overall parity check**, for we see that if $v = (v_1, \ldots, v_{n+1})$ then $v \in \widehat{C}$ satisfies $\sum v_i = 0$. If C has generator matrix G and check matrix H, then $\widehat{C}$ has generator matrix $\widehat{G}$ and check matrix $\widehat{H}$, where $\widehat{G}$ is obtained from G by adding an $(n+1)^{\text{th}}$ column such that the sum of the columns of $\widehat{G}$ is the zero column, and $\widehat{H}$ is obtained from H by attaching an $(n-k+1)^{\text{th}}$ row and $(n+1)^{\text{th}}$ column onto H, the row being all 1's and the column being $(0, 0, \ldots, 0, 1)^t$. If C is binary with d odd, then $\widehat{C}$ will be an $[n+1, k, d+1]$ code.

Example 14.3 Extending the $[7,4,3]$ binary Hamming code gives an $[8,4,4]$ binary code, which is self-dual.

An inverse process to extending an $[n,k,d\,]$ code is that of **puncturing**, which is achieved by simply deleting a coordinate, thus producing a linear code of length $n-1$. The dimension will be k or $k-1$, clearly, but in the great majority of cases the dimension will remain k; the minimum weight may change in either way, but unless the minimum weight of the original code is 1, the minimum weight of the punctured code will be either $d-1$ or d and in the great majority of cases $d-1$.

Another way to obtain new codes is by **shortening**: given an $[n,k,d]$ q-ary code C, for any integer $r \leq k$, we take the subspace C' of all codewords having 0 in a fixed set of r coordinate positions, and then remove those coordinate positions to obtain a code of length $n-r$. For example, if G is a generator matrix for C in standard form, shortening by the first coordinate will clearly produce an $[n-1,k-1,d']$ code, where $d' \geq d$. In this way we obtain $[n-r,k-r,d']$ codes. This techinique is used in Section 14.5.

14.3 Computer memories and Hamming codes

14.3.1 Introduction

Before going on to defining the actual codes, we describe briefly how the memory chips are designed, and how codes may be used, and why they are needed. A fuller account of the use of error-correcting codes in computer memories may be found in the article [11].

The memories of computers are built from silicon chips. Although any one of these chips is reliable, when many thousands are combined in a memory, some might fail. The use of an error-correcting code can mean that a failed chip will be detected and the error corrected. The errors in a chip might occur in the following way: a memory chip is a square array of data-storage cells, for example a $64K$ chip, where $K = 2^{10}$. The $64K$ chip stores $64K = 2^{16} = 65,536$ bits, i.e. binary digits, of data. Alternatively, there are $256K = 2^{18}$ and one-megabit (2^{20} bits) chips. In the $64K$ chip the data-storage cells are arranged in a $2^8 \times 2^8$ array, where each cell stores a 0 or a 1. Each cell can be accessed individually, and has an “address” corresponding to the row and column coordinates, usually numbered from 0 to 255. The largest address is in position $(255,255) = (2^8-1, 2^8-1)$; the binary representation of 255 is 11111111, a sequence of eight bits, and thus the row and column addresses for a $64K$ require eight bits each, i.e. 16 in all. A $256K$ chip has $2^9 = 512$ rows and columns and thus requires 18 bits to address a cell, and a one-megabit chip requires 20.

The 0’s or 1’s stored in a memory chip are represented by the presence or absence of negative electric charges at sites in the silicon crystal whose electrical properties make them potential wells for negative charge. When a 0 is to be stored in a cell the potential well at the site is filled with electrons; when a 1 is to be stored the well is emptied. The cell is read by measuring its negative charge; if the charge is higher than a certain value it is read to be a 0, otherwise a 1.

Clearly, if a potential well lost its charge it would be read incorrectly. Errors do occur: **hard errors** occur when the chip itself is damaged; **soft errors** occur when the chip itself is not damaged but alpha particle bombardment occurs and changes the charge in a cell. The latter is a common cause of error and cannot be avoided.

An error-correcting code is used to correct such errors in the following way: suppose we have a one megabyte memory consisting of 128 64K chips. In such a memory the chips are arranged in four rows of 32 chips each; each chip contains 2^{16} memory cells, so the memory has a total of 2^{23} cells. The data are divided into words of 32 bits each, and each word consists of the contents of one memory cell in each of the 32 chips in one row. In order to correct errors, a further seven chips are added to each of the four rows, making 156 chips. Each row has now 39 chips, the seven extra bit being the parity bits, and are reserved for error-correction. The code actually employed is the binary extended Hamming code of length 64, i.e. a $[64, 57, 4]$ binary code. Such a code will actually protect 57 bits of data, but designers of the codes use only 32 bits.

We now describe how the (binary) Hamming codes are defined.

14.3.2 Hamming codes

These codes were first fully described by Golay [3] and Hamming [4, 5] although the $[7, 4]$ binary code had already appeared in Shannon's fundamental paper. They provide an infinite class of perfect codes. We need here only the binary case, which was the one considered by Hamming. Consider a binary code C with check matrix H: the transposed syndrome, Hy^t, of a received vector y is, in the binary case, simply the sum of those columns of H where the errors occurred. To design a single-error-correcting code we want H not to have any zero columns, since errors in that position would not be detected; similarly, we want H not to have any equal columns, for then errors in those positions would be indistinguishable. If such an H has r rows, n columns and is of rank r, then it will be a check matrix for a single-error-correcting $[n, n-r]$ code. Thus, to maximize the dimension, n should be chosen as large as possible. The number of distinct non-zero r-tuples available for columns is $2^r - 1$. We take for H the $r \times (2^r - 1)$ binary matrix whose columns are all the distinct non-zero r-tuples.

Definition 14.8 *The binary Hamming code of length $2^r - 1$ is the code $\mathcal{H}_r$ that has for check matrix the $r \times (2^r - 1)$ matrix H of all non-zero r-tuples over $\mathbf{F}_2$.*

Theorem 14.3 *The binary code $\mathcal{H}_r$ is a $[2^r - 1, 2^r - 1 - r, 3]$ perfect single-error-correcting code for all $r \geq 2$.*

Example 14.4 (1) If $r = 2$ then

$$H = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

and $\mathcal{H}_2$ is a $[3,1,3]$ code, which is simply the binary repetition code of length 3.

(2) If $r=3$, $\mathcal{H}_3$ is a $[7,4,3]$ binary code, with check matrix

$$H = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Decoding a binary Hamming code is very easy. We first arrange the columns of H, a check matrix for $\mathcal{H}_r$, so that the j^{th} column represents the binary representation (transposed) of the integer j. Now we decode as follows. Suppose the vector y is received. We first find the syndrome, $\text{synd}(y) = yH^t$. If $\text{synd}(y) = 0$, then decode as y, since then $y \in \mathcal{H}_r$. If $\text{synd}(y) \neq 0$, then, assuming one error has occurred, it must have occurred at the j^{th} position where the vector $(\text{synd}(y))^t$ is the binary representation of the integer j. Thus decode y to $y + e_j$, where e_j is the vector of length $n = 2^r - 1$ with 0 in every position except the j^{th}, where it has a 1. The examples given here use this ordering, but notice that the ordering of the entries in the transposed m-tuple representing a number is read from left to right. Thus, [1011] (transposed) represents $1+0+4+8=13$ rather than the customary $1+2+0+8=11$.

If we form the extended binary Hamming code $\widehat{\mathcal{H}_r}$, we obtain a $[2^r, 2^r-1-r, 4]$ code which is still single-error-correcting, but which is capable of simultaneously detecting two errors. This code is useful for *incomplete decoding*: see Hill [6].

Example 14.5 Let $C = \widehat{\mathcal{H}_3}$, with check matrix $\widehat{H}$ obtained as described above (see after Definition 14.7):

$$\widehat{H} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix},$$

and generator matrix

$$\widehat{G} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

The data set $(1,0,1,0)$ is encoded as $(1,0,1,0)\widehat{G} = (1,0,1,1,0,1,0,0)$. If a single error occurs at the i^{th} position, then the received vector y will have $(\text{synd}(y))^t$ the i^{th} column of $\widehat{H}$ and decoding can be performed. However, if two errors occur, at the i^{th} and j^{th} positions, the $(\text{synd}(y))^t$ will be the sum of the i^{th} and j^{th} columns of $\widehat{H}$, and thus will have 0 as the last entry. Decocoding will thus not take place.

Definition 14.9 *The orthogonal code $\mathcal{H}_r^\perp$ of $\mathcal{H}_r$ is called the binary* **simplex** *code and is denoted by $\mathcal{S}_r$.*

The simplex code $\mathcal{S}_r$ clearly has length $2^r - 1$ and dimension r. The generator matrix is H. It follows that $\mathcal{S}_r$ consists of the zero vector and $2^r - 1$ vectors of weight 2^{r-1}, so that it is a $[2^r - 1, r, 2^{r-1}]$ binary code. Any two codewords are at Hamming distance 2^{r-1} and, if the codewords are placed at the vertices of a unit cube in $2^r - 1$ dimensions, they form a simplex. Now from elementary coding theory (see Section 2) we know that a check matrix for $\widehat{\mathcal{H}_r}$ is H with a column of zeros attached, and then a further row with all entries equal to 1: the code spanned by this matrix, i.e. $\widehat{\mathcal{H}_r}^{\perp}$, is a $[2^r, r+1, 2^{r-1}]$ binary code, and is also a first-order Reed-Muller code, denoted by $\mathcal{R}(1, r)$. It can correct $2^{r-2} - 1$ errors.

Finally now, looking back at the application to computer memories, the code used is $\widehat{\mathcal{H}_6}$, a $[64, 57, 4]$ binary code. A check matrix can easily be constructed in the manner described above. The code will now correct any single error that occurs in the way described above, and will *simultaneously* detect any two errors.

14.4 Photographs in space and Reed-Muller codes

14.4.1 Introduction

Photographs of the planet Mars were first taken by the Mariner series of spacecraft in the '60s and early '70s and the first-order Reed-Muller code of length 32 was used to obtain good quality photographs. The original black and white photographs taken by the earlier Mariners were broken down into 200×200 picture elements. Each element was assigned a binary 6-tuple representing one of 64 possible brightness levels from, say, 000000 for white to 111111 for black. Later this division was made finer by using 700×832 elements, and the quality was increased by encoding the 6-tuples using the $[32, 6, 16]$ binary 7-error correcting Reed-Muller code $\mathcal{R}(1, 5)$ in the way described in Section 14.2. When color photographs were taken, the same code was used simply by using the same photograph through different colored filters. In the Voyager series after the late '70s the binary extended Golay code $\mathcal{G}_{24}$, a $[24, 12, 8]$ code, was used in the color photography.

Later on in the Voyager series of spacecraft a different type of codes, *viz.* convolutional Reed-Solomon codes, was used: see [17]. We describe the Reed-Solomon codes in Section 14.5.2.

14.4.2 First-order Reed-Muller codes

A full account of the Reed-Muller codes, which are all binary codes, can be found in [10] or [1, Chapter 5]. We describe here simply a way to construct the $[32, 6, 16]$ code used for space photography, although in fact we can be more general since we already have the construction from the extended Hamming codes. Thus we have the first-order Reed-Muller code $\mathcal{R}(1, m)$ of length 2^m is the code $\widehat{\mathcal{H}_m}^{\perp}$, i.e. the orthogonal of the extended binary Hamming code. It is, as discussed above, a $[2^m, m+1, 2^{m-1}]$ binary $(2^{m-2} - 1)$-error-correcting code. A 6×32 generator matrix G for $\mathcal{R}(1, 5)$ may thus be constructed as follows: first form a 5×31 matrix G_0 with columns the binary representation of each number between 1 and 31 as a column of length 5; then adjoin a column with all entries zero, and finally add a sixth row with all entries equal to 1. Thus

$$G_0 = \begin{bmatrix} 1 & 0 & 1 & \cdots & 0 & 1 \\ 0 & 1 & 1 & \cdots & 1 & 1 \\ 0 & 0 & 0 & \cdots & 1 & 1 \\ 0 & 0 & 0 & \cdots & 1 & 1 \\ 0 & 0 & 0 & \cdots & 1 & 1 \end{bmatrix}$$

and

$$G = \begin{bmatrix} & & G_0 & & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix}.$$

14.4.3 The binary Golay codes

There are many ways to arrive at the perfect Golay $[23, 12, 7]$ binary 3-error-correcting code $\mathcal{G}_{23}$ and its extension, the $[24, 12, 8]$ binary Golay code $\mathcal{G}_{24}$ that was used in the Voyager spacecraft. We will give a generator matrix, as Golay originally did in [3]. That the code generated by this matrix has the specified properties can be verified very easily, or by consulting any of the references given in Section 14.2, for example [6, Chapter 9].

A generator matrix over $\mathbf{F}_2$ for $\mathcal{G}_{24}$ is

$$G = [I_{12}|B], \tag{14.8}$$

where I_{12} is the 12×12 identity matrix and B is a 12×12 matrix given by

$$B = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & & & \\ \vdots & & A & \\ 1 & & & \end{bmatrix}, \tag{14.9}$$

where A is an 11×11 matrix of 0's and 1's defined in the following way: consider the finite field $\mathbf{F}_{11}$ of order 11, i.e. the eleven remainders modulo 11. The first row of A is labelled by these eleven remainders in order, starting with 0, and placing an entry 1 in the i^{th} position if i is a square modulo 11, and a 0 otherwise. The squares modulo 11 are $\{0, 1, 3, 4, 5, 9\}$, and thus the first row of A is

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

For the remaining rows of A simply cycle this row to the left ten times, to obtain

$$A = \begin{bmatrix} 1&1&0&1&1&1&0&0&0&1&0\\ 1&0&1&1&1&0&0&0&1&0&1\\ 0&1&1&1&0&0&0&1&0&1&1\\ 1&1&1&0&0&0&1&0&1&1&0\\ 1&1&0&0&0&1&0&1&1&0&1\\ 1&0&0&0&1&0&1&1&0&1&1\\ 0&0&0&1&0&1&1&0&1&1&1\\ 0&0&1&0&1&1&0&1&1&1&0\\ 0&1&0&1&1&0&1&1&1&0&0\\ 1&0&1&1&0&1&1&1&0&0&0\\ 0&1&1&0&1&1&1&0&0&0&1 \end{bmatrix}. \qquad (14.10)$$

(This construction is quite general in fact and leads to a class of Hadamard matrices and also to the quadratic residue codes: see [1, Chapters 2,7] for further details.) An effective decoding algorithm for $\mathcal{G}_{24}$ is given in [16, Chapter 4].

The perfect binary Golay code $\mathcal{G}_{23}$ may be obtained from $\mathcal{G}_{24}$ by deleting any coordinate.

14.5 Compact discs and Reed-Solomon codes

14.5.1 Introduction

A full account of the use of Reed-Solomon codes for error-correction in compact discs is given in [7], [16, Chapter 7] or [17].

Sound is stored on a compact disc by dividing it up into small parts and representing these parts by binary data, just as pictures are divided up, as described in Section 14.4. A compact disc is made by sampling sound waves 44,100 times per second, the amplitude measured and assigned a value between 1 and $2^{16} - 1$, given as a binary 16-tuple. In fact, two samples are taken, one for the left and one for the right channel. Each binary 16-tuple is taken to represent two field elements from the Galois field of 2^8 elements, $\mathbf{F}_{2^8}$, and thus each sample produces four symbols from $\mathbf{F}_{2^8}$.

For error-correction the information is broken up into segments called *frames*, where each frame holds 24 data symbols. The code used for error-correction is a *Cross Interleaved Reed-Solomon code* (CIRC) obtained by a process called "cross-interleaving" of two shortened Reed-Solomon codes, as described below. The 24 symbols from $\mathbf{F}_{2^8}$ from six samples are used as information symbols in a (shortened) Reed-Solomon $[28, 24, 5]$ code C_1 over $\mathbf{F}_{2^8}$. Another shortened Reed-Solomon $[32, 28, 5]$ code C_2 also over $\mathbf{F}_{2^8}$ is then used in the interleaving process, which has four additional parity-check symbols. See [7, 16, 17] for a detailed description of this process.

As a result of this interleaving process of error-correction, flaws such as scratches on a disc, producing a train of errors called an "error burst", can be corrected.

We describe now the basic Reed-Solomon class of codes.

14.5.2 Reed-Solomon codes

The Reed-Solomon codes are a class of cyclic q-ary codes of length n dividing $q-1$ that satisfy the Singleton bound, and are thus also MDS codes (see Section 14.2). It would take too long to describe general cyclic codes, but we will nevertheless define these codes as being cyclic and illustrate immediately how a generator matrix and a check matrix may be found.

Let $F = \mathbf{F}_q$ and let $n|(q-1)$. Then F has elements of order n, and we let β be such an element. Pick any number δ such that $2 \le \delta \le n$, and take any number a such that $0 \le a \le q-2$. Then the polynomial

$$g(X) = (X - \beta^{1+a})(X - \beta^{2+a})(X - \beta^{3+a}) \dots (X - \beta^{\delta-1+a}) \tag{14.11}$$

is the generator polynomial of an $[n, n-\delta+1, \delta]$ code over F, a **Reed-Solomon** code. In the special case when $n = q-1$ the code is a **primitive** Reed-Solomon code. We obtain a generator polynomial for the code by first expanding the polynomial $g(X)$ to obtain, writing $d = \delta$ (since δ is indeed the minimum weight),

$$g(X) = g_0 + g_1 X + \dots + g_{d-2}X^{d-2} + X^{d-1} \tag{14.12}$$

where $g_i \in F$ for each i. A generator matrix can then be shown to be

$$G = \begin{bmatrix} g_0 & g_1 & g_2 & \dots & g_{d-1} & 1 & 0 & \dots & 0 \\ 0 & g_0 & g_1 & \dots & g_{d-2} & g_{d-1} & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & g_0 & \dots & \dots & g_{d-2} & g_{d-1} & 1 \end{bmatrix}. \tag{14.13}$$

The general theory of BCH codes (see [1, Chapter 2]) immediately gives a check matrix

$$H = \begin{bmatrix} 1 & \beta & \beta^2 & \dots & \beta^{(n-1)} \\ 1 & \beta^2 & \beta^4 & \dots & \beta^{2(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \beta^{d-1} & \beta^{2(d-1)} & \dots & \beta^{(n-1)(d-1)} \end{bmatrix}, \tag{14.14}$$

taking here $a = 0$. This is a convenient generator matrix for the orthogonal code (which is also a Reed-Solomon code) with a rather simple encoding rule: the data set $(a_1, a_2, \dots, a_{d-1})$ is encoded as $(a(1), a(\beta), a(\beta^2), \dots, a(\beta^{n-1}))$ where

$$a(X) = a_1 X + a_2 X^2 \dots a_{d-1} X^{d-1}.$$

The theory (see [1, Chapter 2]) also tells us that the reciprocal polynomial of $h(X) = (X^n - 1)/g(X)$, i.e. $\bar{h}(X) = X^{n-d+1}h(X^{-1})$, is a generator polynomial for the orthogonal code, so another check matrix may be obtained for this in the same way as we did for the code with $g(X)$ as generator polynomial. A more convenient method for the purposes here is to reduce G to standard form and then simply use the formula shown in Equation (14.6).

Example 14.6 Let $q = 11$ and $n = 5$. Then we can take $\beta = 4$ as this has order precisely 5, and let $a = 0$ and $\delta = 3$. Then

$$g(X) = (X - \beta)(X - \beta^2) = (X - 4)(X - 5) = 9 + 2X + X^2$$

so that

$$G = \begin{bmatrix} 9 & 2 & 1 & 0 & 0 \\ 0 & 9 & 2 & 1 & 0 \\ 0 & 0 & 9 & 2 & 1 \end{bmatrix}.$$

Since $(X^5 - 1) = (X - 1)(X - 4)(X - 5)(X - 9)(X - 3)$, $(X^5 - 1)/g(X) = (X - 1)(X - 3)(X - 9)$ and the orthogonal code has generator polynomial

$$\bar{h}(X) = X^3(X^{-1} - 1)(X^{-1} - 3)(X^{-1} - 9) = 1 + 9X + 6X^2 + 6X^3,$$

so that a check matrix is

$$H = \begin{bmatrix} 1 & 9 & 6 & 6 & 0 \\ 0 & 1 & 9 & 6 & 6 \end{bmatrix}.$$

Alternatively, the check matrix can be obtained from Equation (14.14), giving

$$H' = \begin{bmatrix} 1 & 4 & 5 & 9 & 3 \\ 1 & 5 & 3 & 4 & 9 \end{bmatrix},$$

which is of course row-equivalent to H.

The codes used are actually **shortened** Reed-Solomon codes (see Section 14.2). The easiest way to describe these codes is to have a generator matrix of the original Reed-Solomon code C in standard form, which is possible without even having to take an isomorphic code in this case, due to the Reed-Solomon codes having the property of being MDS codes. Thus a generator matrix can be row reduced to standard form, without the need of column operations. (Sometimes it is more convenient to use the standard form for the orthogonal code rather, and thus have the generator matrix in the form $[A|I_k]$.) If now C, an $[n, k, d]$ q-ary code with $n - k = d - 1$, is to be shortened in the first r places to obtain a code C' of length $n - r$, we obtain a generator matrix G' for C' also in standard form by simply deleting the first r rows and columns of G. If H is the check matrix in standard form for G, then the check matrix in standard form for G' is H' obtained from H by deleting the first r columns. It is easy to see that the MDS properties of the original code show that C' is also MDS and is an $[n - r, k - r, d]$ q-ary code, and, of course, $n - r = k - r + d - 1$.

The finite field used in the codes used for compact discs is the field $F = \mathbf{F}_{2^8}$ of order 2^8. This can be constructed as the set of all polynomials over the field $\mathbf{F}_2$ of degree at most 7, i.e.

$$F = \{a_0 + a_1X + a_2X^2 + a_3X^3 + a_4X^4 + a_5X^5 + a_6X^6 + a_7X^7 : a_i \in \mathbf{F}_2\}.$$

Addition of these polynomials is just the standard addition, as is multiplication, except that multiplication is carried out modulo the polynomial

$$1 + X^2 + X^3 + X^4 + X^8. \tag{14.15}$$

Thus $X^8 = 1 + X^2 + X^3 + X^3$ and all other powers of X are reduced to be at most 7, using this rule. The elements of $F = \mathbf{F}_{2^8}$ are thus effectively 8-tuples of binary digits, and this is how they are treated for the application.

The length of the Reed-Solomon codes over F are divisors of $2^8 - 1 = 255 = 3 \times 5 \times 17$. The code actually used is the shortened primitive one, i.e. the shortened $[255, 251, 5]$ code over F, with certain information symbols set to zero. Then if ω is a primitive element for the field (i.e. an element of multiplicative order 255), we take $\beta = \omega$. Since we want minimum distance 5, we take $k = n - 4 = 251$, and shorten to length 28 for C_1 and to length 32 for C_2. The original Reed-Solomon of length 255 would be the same for the two codes, and would have generator polynomial

$$g(X) = (X - \omega)(X - \omega^2)(X - \omega^3)(X - \omega^4) = \omega^{10} + \omega^{81}X + \omega^{251}X^2 + \omega^{76}X^3 + X^4.$$

The powers of ω are then given as binary 8-tuples: for example,

$$\omega^{10} = \omega^2 + \omega^4 + \omega^5 + \omega^6 = (0, 0, 1, 0, 1, 1, 1, 0),$$

and

$$\omega^{251} = \omega^3 + \omega^4 + \omega^6 + \omega^7 = (0, 0, 0, 1, 1, 0, 1, 1),$$

as can be computed using, for example, the computer package Magma [2].

For further details of the properties of finite fields, the reader might consult [8].

14.6 Conclusion

We have been brief in this outline of the use of well-known mathematical constructions in important practical examples, and have not included full details of the implementation of the codes. The reader is urged to consult the bibliography included here for a full and detailed description of the usage. We hope merely to have given some idea as to the nature of the codes, and where they are applied.

The exercises included in the next section should be accessible using the definitions described in this chapter. They are mostly quite elementary. The projects described are open problems whose solution might prove to have rather important applications; some computational results might first be done to make the questions more precise, in particular in the case of Project 2. Magma [2] is a good computational tool to use for this type of problem.

14.7 Exercises and projects

EXERCISES

1. Prove Theorem 14.1.

2. If C is a code of packing radius ρ, show that the spheres with radius ρ and centers the codewords of C do not intersect. Hence deduce the sphere packing bound, Equation 14.2.

3. Prove Theorem 14.2.

4. Let C be a binary code with generator matrix

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

Show that $d(C) = 3$ and construct a check matrix for C. If a vector $y = (0, 1, 1, 1, 1)$ were received, what would you correct it to?

5. State the Singleton bound and the sphere packing bound for an (n, M, d) q-ary code. Compare these to find the best bound for M for a $(10, M, 5)$ binary code.

6. A linear code C is *self-orthogonal* if $C \subseteq C^{\perp}$. Let C be a binary code. Show that

   (a) if C is self-orthogonal then every codeword has even weight;

   (b) if every codeword of C has weight divisible by 4 then C is self-orthogonal;

   (c) if C is self-orthogonal and generated by a set of vectors of weight divisible by 4, then every codeword has weight divisible by 4 (i.e. C is *doubly-even*).

7. Let C be a linear code with check matrix H and covering radius r. Show that r is the maximum weight of a coset leader for C, and that r is the smallest number s such that every syndrome is a linear combination of s or fewer columns of H.

8. Prove from Definition 14.8 that $\mathcal{H}_r$ is a $[2^r - 1, 2^r - 1 - r, 3]$ perfect binary code.

9. Use the diagram shown in Figure 14.2 to decode the received vectors 1101101 and 1001011.

10. Write down a check matrix for $\mathcal{H}_4$, writing the columns so that the j^{th} column is the binary representation of the number j. Use the method described on page 11 to decode $\sum_{i=1}^{10} e_i$ and $\sum_{j=1}^{5} e_{2j-1}$, where e_i is the vector of length 15 with an entry 1 in the i^{th} position, and zeros elsewhere.

11. Show that a binary $[23, 12, 7]$ code is perfect, 3-error-correcting.

12. In the matrix A of Equation (14.10) show that the sum of the first row with any other row has weight 6, and hence that the sum of any distinct two rows of A has weight 6. What can then be said for the sum of any two rows of B?

13. If C is an $[n, k, d]$ code with generator matrix G in standard form, show that shortening of C by the first r coordinate positions, where $r \leq k$, produces an $[n - r, k - r, d']$ code C' where $d' \geq d$. (**Hint:** look at the check matrix for C' and use Theorem 14.2.) Show that it is possible for $d' > d$ by constructing an example.

14. Construct a primitive $[6,4,3]$ Reed-Solomon code C over $\mathbf{F}_7$ using the primitive element 3. Form a shortened code of length 4 and determine its minimum distance.

    Extend C to $\widehat{C}$ (see Definition 14.7). Is $\widehat{C}$ also MDS? Give generator matrices for $\widehat{C}$ and $(\widehat{C})^{\perp}$.

15. Construct the Reed-Solomon code of length 16 and minimum distance 5 by giving its generator polynomial and a check matrix.

PROJECTS

1. Let C be a Hamming code $\mathcal{H}_r$. Is it possible to construct a generator matrix G for C such that every row of G has **exactly** three non-zero entries, and every column of G has **at most** three non-zero entries?
2. Is it possible, in general, to construct a basis of **minimum weight** vectors for other Hamming and Reed-Muller codes?

Bibliography

[1] E. F. Assmus, Jr. and J. D. Key. *Designs and their Codes.* Cambridge Tracts in Mathematics, Vol. 103. Cambridge University Press, Cambridge, 1992 (Second printing with corrections, 1993).

[2] Wieb Bosma and John Cannon. *Handbook of Magma Functions.* Department of Mathematics, University of Sydney, November 1994.

[3] Marcel J. E. Golay. Notes on digital coding. *Proc. IRE*, 37:657, 1949.

[4] R. W. Hamming. Error detecting and error correcting codes. *Bell System Tech. J.*, 29:147–160, 1950.

[5] Richard W. Hamming. *Coding and Information Theory.* Prentice Hall, Englewood Cliffs, N.J., 1980.

[6] Raymond Hill. *A First Course in Coding Theory.* Oxford Applied Mathematics and Computing Science Series. Oxford University Press, Oxford, 1986.

[7] H. Hoeve, J. Timmermans, and L. B. Vries. Error correction and concealment in the compact disc system. *Philips Tech. Rev.*, 40:166–172, 1982.

[8] Rudolf Lidl and Harald Niederreiter. *Introduction to Finite Fields and their Applications.* Cambridge University Press, Cambridge, 1986.

[9] J. H. van Lint. *Introduction to Coding Theory.* Graduate Texts in Mathematics 86. Springer-Verlag, New York, 1982.

[10] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes.* North-Holland, Amsterdam, 1983.

[11] Robert J. McEliece. The reliability of computer memories. *Scientific American*, 252:2–7, 1985.

[12] W. Wesley Peterson. Error-correcting codes. *Scientific American*, 206:96–108, 1962.

[13] Vera Pless. *The Theory of Error Correcting Codes.* Wiley Interscience Series in Discrete Mathematics and Optimization. John Wiley and Sons, New York, 1989 (Second Edition.

[14] C. E. Shannon. A mathematical theory of communication. *Bell System Tech. J.*, 27:379–423,623–656, 1948.

[15] David Slepian. Some further theory of group codes. *Bell System Tech. J.*, 39:1219–1252, 1960.

[16] Scott A. Vanstone and Paul C. van Oorschot. *An Introduction to Error Correcting Codes with Applications.* Kluwer Academic Publishers, 1992.

[17] Stephen B. Wicker and Vijay K. Bhargava (Editors). *Reed-Solomon Codes and their Applications.* IEEE Press, 1994.

April 22, 2003