NORTH-HOLLAND
MATHEMATICAL LIBRARY

The Theory of Error-Correcting Codes

F.J. MACWILLIAMS
N.J.A. SLOANE

North-Holland

The Theory of Error-Correcting Codes

North-Holland Mathematical Library

VOLUME 16

north-holland publishing company
amsterdam · new york · oxford

The Theory of Error-Correcting Codes

F.J. MacWilliams
N.J.A. Sloane

Bell Laboratories
Murray Hill
NJ 07974
U.S.A.

north-holland publishing company
amsterdam · new york · oxford

LCC Number: 76-41296
ISBN: 0 444 85009 0
and 0 444 85010 4

Published by:
North-Holland Publishing Company
Amsterdam · New York · Oxford

Sole distributors for the U.S.A. and Canada:

Elsevier/North-Holland Inc.
52 Vanderbilt Avenue
New York, NY 10017

Library of Congress Cataloging in Publication Data

MacWilliams, Florence Jessie, 1917–
The theory of error-correcting codes.

1. Error-correcting codes (Information theory)
I. Sloane, Neil James Alexander, 1939– joint author.
II. Title.
QA268.M3 519.4 76-41296
ISBN 0 444 85009 0
and 0 444 85010 4

Second printing 1978
Third printing 1981

PRINTED IN THE NETHERLANDS

Preface

Coding theory began in the late 1940's with the work of Golay, Hamming and Shannon. Although it has its origins in an engineering problem, the subject has developed by using more and more sophisticated mathematical techniques. It is our goal to present the theory of error-correcting codes in a simple, easily understandable manner, and yet also to cover all the important aspects of the subject. Thus the reader will find both the simpler families of codes – for example, Hamming, BCH, cyclic and Reed–Muller codes – discussed in some detail, together with encoding and decoding methods, as well as more advanced topics such as quadratic residue, Golay, Goppa, alternant, Kerdock, Preparata, and self-dual codes and association schemes.

Our treatment of bounds on the size of a code is similarly thorough. We discuss both the simpler results – the sphere-packing, Plotkin, Elias and Garshamov bounds – as well as the very powerful linear programming method and the McEliece–Rodemich–Rumsey–Welch bound, the best asymptotic result known. An appendix gives tables of bounds and of the best codes presently known of length up to 512.

Having two authors has helped to keep things simple: by the time we both understand a chapter, it is usually transparent. Therefore this book can be used both by the beginner and by the expert, as an introductory textbook and as a reference book, and both by the engineer and the mathematician. Of course this has not resulted in a thin book, and so we suggest the following menus:

An elementary first course on coding theory for mathematicians: Ch. 1, Ch. 2 (§6 up to Theorem 22), Ch. 3, Ch. 4 (§§1–5), Ch. 5 (to Problem 5), Ch. 7 (not §§7, 8), Ch. 8 (§§1–3), Ch. 9 (§§1, 4), Ch. 12 (§8), Ch. 13 (§§1–3), Ch. 14 (§§1–3).

A second course for mathematicians: Ch. 2 (§§1–6, 8), Ch. 4 (§§6, 7 and part of 8), Ch. 5 (to Problem 6, and §§3, 4, 5, 7), Ch. 6 (§§1–3, 10, omitting the

proof of Theorem 33), Ch. 8 (§§5, 6), Ch. 9 (§§2, 3, 5), Ch. 10 (§§1–5, 11), Ch. 11, Ch. 13 (§§4, 5, 9), Ch. 16 (§§1–6), Ch. 17 (§7, up to Theorem 35), Ch. 19 (§§1–3).

An elementary first course on coding theory for engineers: Ch. 1, Ch. 3, Ch. 4 (§§1–5), Ch. 5 (to Problem 5), Ch. 7 (not §7), Ch. 9 (§§1, 4, 6), Ch. 10 (§§1, 2, 5, 6, 7, 10), Ch. 13 (§§1–3, 6, 7), Ch. 14 (§§1, 2, 4).

A second course for engineers: Ch. 2 (§§1–6), Ch. 8 (§§1–3, 5, 6), Ch. 9 (§§2, 3, 5), Ch. 10 (§11), Ch. 12 (§§1–3, 8, 9), Ch. 16 (§§1, 2, 4, 6, 9), Ch. 17 (§7, up to Theorem 35).

There is then a lot of rich food left for an advanced course: the rest of Chapters 2, 6, 11 and 14, followed by Chapters 15, 18, 19, 20 and 21 – a feast!

The following are the principal codes discussed:

Alternant, Ch. 12;
BCH, Ch. 3, §§1, 3; Ch. 7, §6; Ch. 8, §5; Ch. 9; Ch. 21, §8;
Chien–Choy generalized BCH, Ch. 12, §7;
Concatenated, Ch. 10, §11; Ch. 18, §§5, 8;
Conference matrix, Ch. 2, §4;
Cyclic, Ch. 7, Ch. 8;
Delsarte–Goethals, Ch. 15, §5;
Difference-set cyclic, Ch. 13, §8;
Double circulant and quasi-cyclic, Ch. 16, §§6–8;
Euclidean and projective geometry, Ch. 13, §8;
Goethals generalized Preparata, Ch. 15, §7;
Golay (binary), Ch. 2, §6; Ch. 16, §2; Ch. 20;
Golay (ternary), Ch. 16, §2; Ch. 20;
Goppa, Ch. 12, §§3–5;
Hadamard, Ch. 2, §3;
Hamming, Ch. 1, §7, Ch. 7, §3 and Problem 8;
Irreducible or minimal cyclic, Ch. 8, §§3, 4;
Justesen, Ch. 10, §11;
Kerdock, Ch. 2, §8; Ch. 15, §5;
Maximal distance separable, Ch. 11;
Nordstrom–Robinson, Ch. 2, §8; Ch. 15, §§5, 6;
Pless symmetry, Ch. 16, §8;
Preparata, Ch. 2, §8; Ch. 15, §6; Ch. 18, §7.3;
Product, Ch. 18, §§2–6;
Quadratic residue, Ch. 16;
Redundant residue, Ch. 10, §9;
Reed–Muller, Ch. 1, §9; Chs. 13–15;
Reed–Solomon, Ch. 10;

Self-dual, Ch. 19;
Single-error-correcting nonlinear, Ch. 2, §7; Ch. 18, §7.3;
Srivastava, Ch. 12, §6.

Encoding methods are given for:

Linear codes, Ch. 1, §2;
Cyclic codes, Ch. 7, §8;
Reed–Solomon codes, Ch. 10, §7;
Reed–Muller codes, Ch. 13, §§6, 7; Ch. 14, §4.

Decoding methods are given for:

Linear codes, Ch. 1, §§3, 4;
Hamming codes, Ch. 1, §7;
BCH codes, Ch. 3, §3; Ch. 9, §6; Ch. 12, §9;
Reed–Solomon codes, Ch. 10, §10;
Alternant (including BCH, Goppa, Srivastava and Chien–Choy generalized BCH codes) Ch. 12, §9;
Quadratic residue codes, Ch. 16, §9;
Cyclic codes, Ch. 16, §9,

while other decoding methods are mentioned in the notes to Ch. 16.

When reading the book, keep in mind this piece of advice, which should be given in every preface: if you get stuck on a section, skip it, but keep reading! Don't hesitate to skip the proof of a theorem: we often do. Starred sections are difficult or dull, and can be omitted on the first (or even second) reading.

The book ends with an extensive bibliography. Because coding theory overlaps with so many other subjects (computers, digital systems, group theory, number theory, the design of experiments, etc.) relevant papers may be found almost anywhere in the scientific literature. Unfortunately this means that the usual indexing and reviewing journals are not always helpful. We have therefore felt an obligation to give a fairly comprehensive bibliography. The notes at the ends of the chapters give sources for the theorems, problems and tables, as well as small bibliographies for some of the topics covered (or not covered) in the chapter.

Only *block* codes for correcting *random* errors are discussed; we say little about codes for correcting other kinds of errors (bursts or transpositions) or about variable length codes, convolutional codes or source codes (see the

Notes to Ch. 1). Furthermore we have often considered only *binary* codes, which makes the theory a lot simpler. Most writers take the opposite point of view: they think in binary but publish their results over arbitrary fields.

There are a few topics which were included in the original plan for the book but have been reluctantly omitted for reasons of space:

(i) Gray codes and snake-in-the-box codes – see Adelson et al. [5, 6], Buchner [210], Cavior [253], Chien et al. [290], Cohn [299], Danzer and Klee [328], Davies [335], Douglas [382, 383], Even [413], Flores [432], Gardner [468], Gilbert [481], Guy [571], Harper [605], Klee [764–767], Mecklenberg et al. [951], Mills [956], Preparata and Nievergelt [1083], Singleton [1215], Tang and Liu [1307], Vasil'ev [1367], Wyner [1440] and Yuen [1448, 1449].

(ii) Comma-free codes – see Ball and Cummings [60, 61], Baumert and Cantor [85], Crick et al. [316], Eastman [399], Golomb [523, pp. 118–122], Golomb et al. [528], Hall [587, pp. 11–12], Jiggs [692], Miyakawa and Moriya [967], Niho [992] and Redinbo and Walcott [1102]. See also the remarks on codes for synchronizing in the Notes to Ch. 1.

(iii) Codes with unequal error protection – see Gore and Kilgus [549], Kilgus and Gore [761] and Mandelbaum [901].

(iv) Coding for channels with feedback – see Berlekamp [124], Horstein [664] and Schalkwijk et al. [1153–1155].

(v) Codes for the Gaussian channel – see Biglieri et al. [148–151], Blake [155, 156, 158], Blake and Mullin [162], Chadwick et al. [256, 257], Gallager [464], Ingemarsson [683], Landau [791], Ottoson [1017], Shannon [1191], Slepian [1221–1223] and Zetterberg [1456].

(vi) The complexity of decoding – see Bajoga and Walbesser [59], Chaitin [257a–258a], Gelfand et al. [471], Groth [564], Justesen [706], Kolmogorov [774a], Marguinaud [916], Martin-Löf [917a], Pinsker [1046a], Sarwate [1145] and Savage [1149–1152a].

(vii) The connections between coding theory and the packing of equal spheres in n-dimensional Euclidean space – see Leech [803–805], [807], Leech and Sloane [808–810] and Sloane [1226].

The following books and monographs on coding theory are our predecessors: Berlekamp [113, 116], Blake and Mullin [162], Cameron and Van Lint [234], Golomb [522], Lin [834], Van Lint [848], Massey [922a], Peterson [1036a], Peterson and Weldon [1040], Solomon [1251] and Sloane [1227a]; while the following collections contain some of the papers in the bibliography: Berlekamp [126], Blake [157], the special issues [377a, 678, 679], Hartnett [620], Mann [909] and Slepian [1224]. See also the bibliography [1022].

We owe a considerable debt to several friends who read the first draft very carefully, made numerous corrections and improvements, and frequently saved us from dreadful blunders. In particular we should like to thank I.F. Blake, P. Delsarte, J.-M. Goethals, R.L. Graham, J.H. van Lint, G. Longo, C.L. Mallows, J. McKay, V. Pless, H.O. Pollak, L.D. Rudolph, D.W. Sarwate, many other colleagues at Bell Labs, and especially A.M. Odlyzko for

their help. Not all of their suggestions have been followed, however, and the authors are fully responsible for the remaining errors. (This conventional remark is to be taken seriously.) We should also like to thank all the typists at Bell Labs who have helped with the book at various times, our secretary Peggy van Ness who has helped in countless ways, and above all Marion Messersmith who has typed and retyped most of the chapters. Sam Lomonaco has very kindly helped us check the galley proofs.

This page intentionally left blank

Preface to the third printing

We should like to thank many friends who have pointed out errors and misprints. The corrections have either been made in the text or are listed below.

A Russian edition was published in 1979 by Svyaz (Moscow), and we are extremely grateful to L. A. Bassalygo, I. T. Grushko and V. A. Zinov'ev for producing a very careful translation. They supplied us with an extensive list of corrections. They also point out (in footnotes to the Russian edition) a number of places we did not cite the earliest source for a theorem. We have corrected the most glaring omissions, but future historians of coding theory should also consult the Russian edition.

Problem 17, page 75. Shmuel Schreiber has pointed out that not all ways of choosing the matrices A, B, C, D work. One choice which does work is

$$A = \begin{bmatrix} 0001 \\ 1000 \\ 0010 \\ 0100 \end{bmatrix}, \quad B = \begin{bmatrix} 0010 \\ 0100 \\ 0001 \\ 1000 \end{bmatrix}, \quad C = \begin{bmatrix} 0100 \\ 0010 \\ 1000 \\ 0001 \end{bmatrix}, \quad D = \begin{bmatrix} 1000 \\ 0001 \\ 0100 \\ 0010 \end{bmatrix}.$$

Page 36, Notes to §2. Add after Wu [1435, 1436]: K. Sh. Zigangirov, Some sequential decoding procedures, *Problems of Information Transmission*, **2** (4) (1966) 1–10; and K. Sh. Zigangirov, *Sequential Decoding Procedures* (Svyaz, Moscow, 1974).

Page 72, Research problem 2.4. It is now known that $A(10, 4) = 40$, $72 \leq A(11, 4) \leq 79$, and $144 \leq A(12, 4) \leq 158$. See M. R. Best, Binary codes with a minimum distance of four, *IEEE Trans. Info. Theory*, Vol. **IT-26** (6) (November 1980), 738–743.

Page 123, Research problem 4.1. Self-contained proofs have been given by O. Moreno, On primitive elements of Trace 1 in GF(2^m), *Discrete Math.*, to appear, and L. R. Vermani, Primitive elements with nonzero trace, preprint.

Chapter 6, pp. 156 and 180. Theorem 33 was proved independently by Zinov'ev and Leont'ev [1472].

Page 166, Research problem 6.1 has been settled in the affirmative by I. I. Dumer, A remark about codes and tactical configurations, *Math. Notes*, to appear; and by C. Roos, Some results on t-constant codes, *IEEE Trans. Info. Theory*, to appear.

Page 175. Research problem 6.3 has also been solved by Dumer and Roos [op. cit.].

Page 178–179. Research problems 6.4 and 6.5 have been solved by Dumer [op. cit.]. The answer to 6.5 is No.

Page 267, Fig. 9.1. R. E. Kibler (private communication) has pointed out that the asterisks may be removed from the entries [127, 29, 43], [255, 45, 87] and [255, 37, 91], since there are minimum weight codewords which are low-degree multiples of the generator polynomial.

Page 280, Research problem 9.4. T. Helleseth (*IEEE Trans. Info. Theory*, Vol. **IT-25** (1979) 361–362) has shown that no other binary primitive BCH codes are quasi-perfect.

Page 299, Research problem 10.1. R. E. Kibler (private communication) has found a large number of such codes.

Page 323. The proof of Theorem 9 is valid only for $q = 2^m$. For odd q the code need not be cyclic. See G. Falkner, W. Heise, B. Kowol and E. Zehender, On the existence of cyclic optimal codes, *Atti Sem. Mat. Fis. Università di Modena*, to appear.

Page 394, line 9 from the bottom. As pointed out by Massey in [918, p. 100], the number of majority gates required in an L-step decoder need never exceed the dimension of the code.

Page 479. Research problem 15.2 is also solved in I. I. Dumer, Some new uniformly packed codes, in: *Proceedings MFTI*, Radiotechnology and Electronics Series (MFTI, Moscow, 1976) pp. 72–78.

Page 546. The answer to Research problem 17.6 is No, and in fact R. E. Kibler, Some new constant weight codes, *IEEE Trans. Info. Theory*, Vol. **IT-26** (May 1980) 364–365, shows that $27 \leq A(24, 10, 8) \leq 68$.

Appendix A, Figures 1 and 3. For later versions of the tables of $A(n, d)$ and $A(n, d, w)$ see R. L. Graham and N. J. A. Sloane, Lower bounds for constant weight codes, *IEEE Trans. Info. Theory*, Vol. **IT-26** (1980) 37–43; M. R. Best, Binary codes with a minimum distance of four, loc. cit., Vol. IT-26 (1980), 738–743; and other papers in this journal.

On page 682, line 6, the value of X corresponding to $F = 30$ should be changed from .039 to .093.

Contents

Chapter 3. An introduction to BCH codes and finite fields

Chapter 4. Finite fields

Chapter 5. Dual codes and their weight distribution

Chapter 6. Codes, designs and perfect codes

Chapter 7. Cyclic codes

Chapter 8. Cyclic codes (contd.): Idempotents and Mattson–Solomon polynomials

Chapter 9. BCH codes

Chapter 16. Quadratic-residue codes

Chapter 17. Bounds on the size of a code

Chapter 18. Methods for combining codes

Appendix A. Tables of the best codes known

Appendix B. Finite geometries

Part I

This page intentionally left blank

1

Linear codes

§1. Linear codes

Codes were invented to correct errors on noisy communication channels. Suppose there is a telegraph wire from Boston to New York down which 0's and 1's can be sent. Usually when a 0 is sent it is received as a 0, but occasionally a 0 will be received as a 1, or a 1 as a 0. Let's say that on the average 1 out of every 100 symbols will be in error. I.e. for each symbol there is a probability $p = 1/100$ that the channel will make a mistake. This is called a *binary symmetric channel* (Fig. 1.1).

There are a lot of important messages to be sent down this wire, and they must be sent as quickly and reliably as possible. The messages are already written as a string of 0's and 1's – perhaps they are being produced by a computer.

We are going to *encode* these messages to give them some protection against errors on the channel. A block of k message symbols $\boldsymbol{u} = u_1 u_2 \ldots u_k$

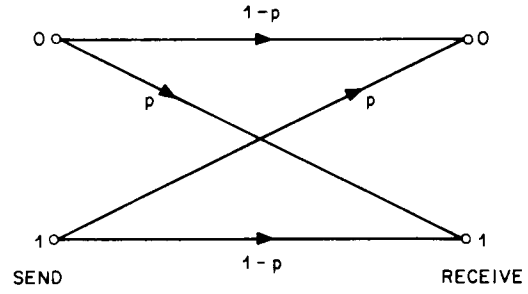

Fig. 1.1. The binary symmetric channel, with error probability p. In general $0 \leqslant p \leqslant \frac{1}{2}$.

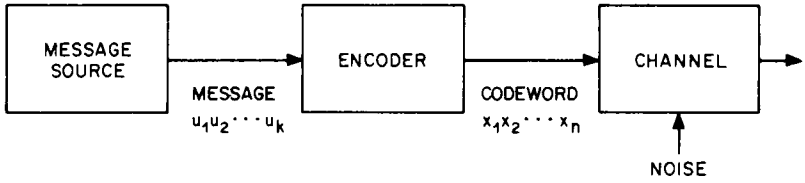

Fig. 1.2

($u_i = 0$ or 1) will be encoded into a *codeword* $\boldsymbol{x} = x_1x_2 \ldots x_n$ ($x_i = 0$ or 1) where $n \geqslant k$ (Fig. 1.2); these codewords form a *code.*

The method of encoding we are about to describe produces what is called a *linear* code. The first part of the codeword consists of the message itself:

$$x_1 = u_1, \quad x_2 = u_2, \quad \ldots, \quad x_k = u_k,$$

followed by $n - k$ *check* symbols

$$x_{k+1}, \ldots, x_n.$$

The check symbols are chosen so that the codewords satisfy

$$H\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = H\boldsymbol{x}^{\mathrm{tr}} = 0, \tag{1}$$

where the $(n-k) \times n$ matrix H is the *parity check matrix* of the code, given by

$$H = [A|I_{n-k}], \tag{2}$$

A is some fixed $(n-k) \times k$ matrix of 0's and 1's, and

$$I_{n-k} = \begin{pmatrix} 1 & & 0 \\ & 1 & \\ 0 & & 1 \end{pmatrix}$$

is the $(n-k) \times (n-k)$ unit matrix. The arithmetic in Equation (1) is to be performed *modulo 2*, i.e. $0 + 1 = 1$, $1 + 1 = 0$, $-1 = +1$. We shall refer to this as *binary arithmetic.*

Example. *Code # 1.* The parity check matrix

$$H = \left[\begin{array}{ccc|ccc} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right] \tag{3}$$

defines a code with $k = 3$ and $n = 6$. For this code

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

The message $u_1u_2u_3$ is encoded into the codeword $\boldsymbol{x} = x_1x_2x_3x_4x_5x_6$, which begins with the message itself:

$$x_1 = u_1, \quad x_2 = u_2, \quad x_3 = u_3,$$

followed by three check symbols $x_4x_5x_6$ chosen so that $H\boldsymbol{x}^{tr} = 0$, i.e. so that

$$\begin{aligned} x_2 + x_3 + x_4 &= 0, \\ x_1 + x_3 + x_5 &= 0, \\ x_1 + x_2 + x_6 &= 0. \end{aligned} \tag{4}$$

If the message is $\boldsymbol{u} = 011$, then $x_1 = 0$, $x_2 = 1$, $x_3 = 1$, and the check symbols are

$$\begin{aligned} x_4 &= -1 - 1 = 1 + 1 = 2 = 0, \\ x_5 &= -1 = 1, \qquad x_6 = -1 = 1, \end{aligned}$$

so the codeword is $x = 011011$.

The Equations (4) are called the *parity check equations*, or simply *parity checks*, of the code.

The first parity check equation says that the 2nd, 3rd and 4th symbols of every codeword must add to 0 modulo 2; i.e. their sum must have even parity (hence the name!).

Since each of the 3 message symbols $u_1u_2u_3$ is 0 or 1, there are altogether $2^3 = 8$ codewords in this code. They are:

000000	011011	110110
001110	100011	111000.
010101	101101	

In the general code there are 2^k codewords.

As we shall see, code # 1 is capable of correcting a single channel error (in any one of the six symbols), and using this code reduces the average probability of error per symbol from $p = .01$ to .00072 (see Problem 24). This is achieved at the cost of sending 6 symbols only 3 of which are message symbols.

We take (1) as our general definition:

Definition. Let H be any binary matrix. The *linear code* with *parity check matrix* H consists of all vectors $\boldsymbol{x}$ such that

$$H\boldsymbol{x}^{tr} = 0.$$

(where this equation is to be interpreted modulo 2).

It is convenient, but not essential, if H has the form shown in (2) and (3), in which case the first k symbols in each codeword are *message* or *information* symbols, and the last $n - k$ are *check* symbols.

Linear codes are the most important for practical applications and are the simplest to understand. Nonlinear codes will be introduced in Ch. 2.

Example. *Code # 2, a repetition code.* A code with $k = 1$, $n = 5$, and parity check matrix

$$H = \begin{bmatrix} 1 & 1 & & & \\ 1 & & 1 & & \\ 1 & & & 1 & \\ 1 & & & & 1 \end{bmatrix} \quad \text{(blanks denote zeros).}$$

Each codeword contains just one message symbol u. The parity check equations are

$$x_1 + x_2 = 0, \qquad x_1 + x_3 = 0, \qquad x_1 + x_4 = 0, \qquad x_1 + x_5 = 0,$$

i.e. $x_1 = x_2 = x_3 = x_4 = x_5 = u$. So there are only two codewords, 00000 and 11111. The message symbol is simply repeated 5 times: this is called a *repetition* code.

Example. *Code # 3, an even weight code.* A code with $k = 3$, $n = 4$ and parity check matrix $H = (1111)$. Each codeword contains 3 message symbols $x_1x_2x_3$ and one check symbol $x_4 = x_1 + x_2 + x_3$. The $2^3 = 8$ codewords are 0000, 0011, 0101, 1001, 0110, 1010, 1100, 1111, i.e. all vectors with an even number of 1's.

Problems. (1) *Code # 4* has parity check matrix

$$H = \begin{bmatrix} 1010 \\ 1101 \end{bmatrix}.$$

List all the codewords. Repeat for

$$H = \begin{bmatrix} 0111 \\ 1101 \end{bmatrix}.$$

How are these codes related?

(2) *Code # 5* has parity check matrix

$$H = \begin{bmatrix} 0111100 \\ 1011010 \\ 1101001 \end{bmatrix}.$$

List all the codewords.

(3) If $p > \frac{1}{2}$ in Fig. 1.1, show that interchanging the names of the received symbols changes this to a binary symmetric channel with $p < \frac{1}{2}$. If $p = \frac{1}{2}$ show that no communication is possible.

§2. Properties of a linear code

(i) The definition again: $x = x_1 \cdots x_n$ is a codeword if and only if

$$Hx^{tr} = 0. \tag{1}$$

(ii) Usually the parity check matrix H is an $(n-k) \times n$ matrix of the form

$$H = [A \mid I_{n-k}], \tag{2}$$

and as we have seen there are 2^k codewords satisfying (1). (This is still true even if H doesn't have this form, provided H has n columns and $n-k$ linearly independent rows.) When H has the form (2), the codewords look like this:

$$x = \underbrace{x_1 \cdots x_k}_{\substack{\text{message} \\ \text{symbols}}} \quad \underbrace{x_{k+1} \cdots x_n}_{\substack{\text{check} \\ \text{symbols}}}.$$

(iii) *The generator matrix.* If the message is $u = u_1 \cdots u_k$, what is the corresponding codeword $x = x_1 \cdots x_n$? First $x_1 = u_1, \ldots, x_k = u_k$, or

$$\begin{pmatrix} x_1 \\ \vdots \\ x_k \end{pmatrix} = I_k \begin{pmatrix} u_1 \\ \vdots \\ u_k \end{pmatrix}, \quad I_k = \text{unit matrix}. \tag{5}$$

Then from (1) and (2),

$$[A \mid I_{n-k}] \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = 0,$$

$$\begin{pmatrix} x_{k+1} \\ \vdots \\ x_n \end{pmatrix} = -A \begin{pmatrix} x_1 \\ \vdots \\ x_k \end{pmatrix}$$

$$= -A \begin{pmatrix} u_1 \\ \vdots \\ u_k \end{pmatrix} \quad \text{from (5)}. \tag{6}$$

In the binary case $-A = A$, but later we shall treat cases where $-A \neq A$. Putting (5) on top of (6):

$$\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{bmatrix} I_k \\ \hline -A \end{bmatrix} \begin{pmatrix} u_1 \\ \vdots \\ u_k \end{pmatrix}$$

and transposing, we get

$$x = uG \tag{7}$$

where

$$G = [I_k \mid -A^{tr}]. \tag{8}$$

If H is in standard form G is easily obtained from H – see (2). G is called a *generator matrix* of the code, for (7) just says that the codewords are all possible linear combinations of the rows of G. (We could have used this as the definition of the code.) (1) and (7) together imply that G and H are related by

$$GH^{tr} = 0 \quad \text{or} \quad HG^{tr} = 0. \tag{9}$$

Example. *Code # 1* (cont.). A generator matrix is

$$G = [I_3 \mid -A^{tr}] = \left[\begin{array}{c|c} 100 & 011 \\ 010 & 101 \\ 001 & 110 \end{array}\right] \begin{array}{l} \text{row 1} \\ \text{row 2} \\ \text{row 3.} \end{array}$$

The 8 codewords are (from (7))

$$u_1 \cdot \text{row } 1 + u_2 \cdot \text{row } 2 + u_3 \cdot \text{row } 3 \quad (u_1, u_2, u_3 = 0 \text{ or } 1).$$

We see once again that the codeword corresponding to the message $\boldsymbol{u} = 011$ is

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{u}G \\ &= \text{row } 2 + \text{row } 3 \\ &= 010101 + 001110 \\ &= 011011, \end{aligned}$$

the addition being done mod 2 as usual.

(iv) *The parameters of a linear code.* The codeword $\boldsymbol{x} = x_1 \cdots x_n$ is said to have *length* n. This is measuring the length as a tailor would, not a mathematician. n is also called the block length of the code. If H has $n - k$ linearly independent rows, there are 2^k codewords. k is called the *dimension* of the code. We call the code an $[n, k]$ code.

This code uses n symbols to send k message symbols, so it is said to have *rate* or *efficiency* $R = k/n$.

(v) *Other generator and parity check matrices.* A code can have several different generator matrices. E.g.

$$\begin{bmatrix} 1110 \\ 0101 \end{bmatrix}, \quad \begin{bmatrix} 1011 \\ 0101 \end{bmatrix}$$

are both generator matrices for code # 4. In fact any maximal set of linearly independent codewords taken from a given code can be used as the rows of a generator matrix for that code.

A *parity check* on a code $\mathscr{C}$ is any row vector $\boldsymbol{h}$ such that $\boldsymbol{h}\boldsymbol{x}^{tr} = 0$ for all codewords $\boldsymbol{x} \in \mathscr{C}$. Then similarly any maximal set of linearly independent parity checks can be used as the rows of a parity check matrix H for $\mathscr{C}$. E.g.

$$\begin{bmatrix} 1010 \\ 1101 \end{bmatrix}, \quad \begin{bmatrix} 0111 \\ 1101 \end{bmatrix}$$

are both parity check matrices for code # 4.

(vi) *Codes over other fields.* Instead of only using 0's and 1's, we could have allowed the symbols to be from any finite field (see Chapters 3 and 4). For example a *ternary* code has symbols 0, 1 and 2, and all calculations of parity checks etc. are done modulo 3 ($1+2=0$, $-1=2$, $2+2=1$, $2\cdot 2=1$, etc.). If the symbols are from a finite field with q elements, the code is still defined by (1) and (2), or equivalently by (7) and (8), and an $[n, k]$ code contains q^k codewords.

Example. *Code # 6.* A [4, 2] ternary code with parity check matrix

$$H = \begin{bmatrix} 1110 \\ 1201 \end{bmatrix} = [A \mid I_2].$$

The generator matrix is (from (8))

$$G = [I_2 \mid -A^{\mathrm{tr}}] = \begin{bmatrix} 1022 \\ 0121 \end{bmatrix} \begin{matrix} \text{row 1} \\ \text{row 2.} \end{matrix}$$

There are 9 codewords $u_1 \cdot \text{row 1} + u_2 \cdot \text{row 2}$ ($u_1, u_2 = 0$, 1 or 2), as follows:

message u	codeword x	message u	codeword x	message u	codeword x
00	0000	10	1022	20	2011
01	0121	11	1110	21	2102
02	0212	12	1201	22	2220

This code has rate $R = k/n = \frac{1}{2}$.

(vii) *Linearity.* If x and y are codewords of a given code, so is $x+y$, because $H(x+y)^{\mathrm{tr}} = Hx^{\mathrm{tr}} + Hy^{\mathrm{tr}} = 0$. If c is any element of the field, then cx is also a codeword, because $H(cx)^{\mathrm{tr}} = cHx^{\mathrm{tr}} = 0$. E.g. in a ternary code if x is a codeword so is $2x = -x$. That is why these are called linear codes. Such a code is also an additive group, and a vector space over the field.

Problems. (4) *Code # 2* (cont.). Give parity check and generator matrices for the general $[n, 1]$ *repetition code.*

(5) *Code # 3* (cont.). Give parity check and generator matrices for the general $[n, n-1]$ *even weight code.*

(6) If the code $\mathscr{C}$ has an invertible generator matrix, what is $\mathscr{C}$?

§3. At the receiving end

(We now return to binary codes.) Suppose the message $u = u_1 \cdots u_k$ is encoded into the codeword $x = x_1 \cdots x_n$, which is then sent through the channel. Because of channel noise, the received vector $y = y_1 \cdots y_n$ may be

different from x. Let's define the *error vector*

$$e = y - x = e_1 \cdots e_n. \tag{10}$$

Then $e_i = 0$ with probability $1-p$ (and the i^{th} symbol is correct), and $e_i = 1$ with probability p (and the i^{th} symbol is wrong). In the example of §1 p was equal to 1/100, but in general p can be anywhere in the range $0 \leq p < \frac{1}{2}$. So we describe the action of the channel by saying it distorts the codeword x by adding the error vector e to it.

The decoder (Fig. 1.3) must decide from y which message u or (usually simpler) which codeword x was transmitted. Of course it's enough if the decoder finds e, for then $x = y - e$. Now the decoder can never be certain what e was. His strategy therefore will be to choose the *most likely* error vector e, given that y was received. Provided the codewords are all equally likely, this strategy is optimum in the sense that it minimizes the probability of the decoder making a mistake, and is called *maximum likelihood decoding.*

To describe how the decoder does this, we need two important definitions.

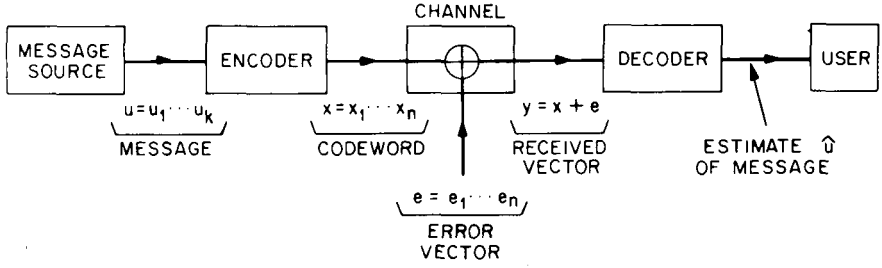

Fig. 1.3. The overall communication system.

Definition. The (*Hamming*) *distance* between two vectors $x = x_1 \cdots x_n$ and $y = y_1 \cdots y_n$ is the number of places where they differ, and is denoted by $\text{dist}(x, y)$. E.g.

$$\text{dist}\,(10111, 00101) = 2, \qquad \text{dist}\,(0122, 1220) = 3$$

(the same definition holds for nonbinary vectors).

Definition. The (*Hamming*) *weight* of a vector $x = x_1 \cdots x_n$ is the number of nonzero x_i, and is denoted by $\text{wt}(x)$. E.g.

$$\text{wt}\,(101110) = 4, \qquad \text{wt}\,(01212110) = 6.$$

Obviously

$$\text{dist}\,(x, y) = \text{wt}\,(x - y), \tag{11}$$

for both sides express the number of places where x and y differ.

Problem. (7) Define the *intersection* of binary vectors x and y to be the vector

$$x * y = (x_1 y_1, \ldots, x_n y_n),$$

which has 1's only where both x and y do. E.g. $11001 * 10111 = 10001$. Show that

$$\mathrm{wt}(x+y) = \mathrm{wt}(x) + \mathrm{wt}(y) - 2\,\mathrm{wt}(x * y). \tag{12}$$

Now back to decoding. Errors occur with probability p. For instance

$$\begin{aligned}
&\mathrm{Prob}\{e = 00000\} = (1-p)^5,\\
&\mathrm{Prob}\{e = 01000\} = p(1-p)^4,\\
&\mathrm{Prob}\{e = 10010\} = p^2(1-p)^3.
\end{aligned}$$

In general if v is some fixed vector of weight a,

$$\mathrm{Prob}\{e = v\} = p^a(1-p)^{n-a}. \tag{13}$$

Since $p < \frac{1}{2}$, we have $1-p > p$, and

$$(1-p)^5 > p(1-p)^4 > p^2(1-p)^3 > \cdots.$$

Therefore a particular error vector of weight 1 is more likely than a particular error vector of weight 2, and so on. So the decoder's strategy is:

Decode y as the nearest codeword x (nearest in Hamming distance), i.e.:

Pick that error vector e which has least weight.

This is called *nearest neighbor decoding.*

A brute force decoding scheme then is simply to compare y with all 2^k codewords and pick the closest. This is fine for small codes. But if k is large this is impossible! One of the aims of coding theory is to find codes which can be decoded by a faster method than this.

The minimum distance of a code. The third important parameter of a code $\mathscr{C}$, besides the length and dimension, is the minimum Hamming distance between its codewords:

$$\begin{aligned}
\mathbf{d} &= \min \mathrm{dist}(u, v)\\
&= \min \mathrm{wt}(u - v) \quad u \in \mathscr{C}, v \in \mathscr{C}, u \neq v.
\end{aligned} \tag{14}$$

d is called the *minimum distance* or simply the *distance* of the code. Any two codewords differ in at least d places.

A linear code of length n, dimension k, and minimum distance d will be called an $[n, k, d]$ code.

To find the minimum distance of a linear code it is not necessary to compare every pair of codewords. For if u and v belong to a linear code $\mathscr{C}$, $u - v = w$ is also a codeword, and (from (14))

$$d = \min_{w \in \mathscr{C}, w \neq 0} \mathrm{wt}(w).$$

In other words:

Theorem 1. *The minimum distance of a linear code is the minimum weight of any nonzero codeword.*

Example. The minimum distances of codes #1, #2, #3, #6 are 3, 5, 2, 3 respectively.

How many errors can a code correct?

Theorem 2. *A code with minimum distance d can correct* $[\frac{1}{2}(d-1)]$ *errors.* If d is even, the code can simultaneously correct* $\frac{1}{2}(d-2)$ *errors and detect d/2 errors.*

Proof. Suppose $d = 3$ (Fig. 1.4). The *sphere* of radius r and center $\boldsymbol{u}$ consists of all vectors $\boldsymbol{v}$ such that $\text{dist}(\boldsymbol{u}, \boldsymbol{v}) \leq r$. If a sphere of radius 1 is drawn around each codeword, these spheres do not overlap. Then if codeword $\boldsymbol{u}$ is transmitted and one error occurs, so that the vector $\boldsymbol{a}$ is received, then $\boldsymbol{a}$ is inside the sphere around $\boldsymbol{u}$, and is still closer to $\boldsymbol{u}$ than to any other codeword $\boldsymbol{v}$. Thus nearest neighbor decoding will correct this error.

Similarly if $d = 2t + 1$, spheres of radius t around each codeword do not overlap, and the code can correct t errors.

Now suppose d is even (Fig. 1.5, where $d = 4$). Spheres of radius $\frac{1}{2}(d-2)$ around the codewords are disjoint and so the code can correct $\frac{1}{2}(d-2)$ errors. But if $d/2$ errors occur the received vector $\boldsymbol{a}$ may be midway between 2 codewords (Fig. 1.5). In this case the decoder can only detect that $d/2$ (or more) errors have occurred. Q.E.D.

Thus code #1, which has minimum distance 3, is a single-error-correcting code.

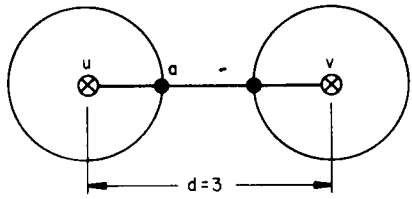

Fig. 1.4. A code with minimum distance 3 (⊗=codeword).

*$[x]$ denotes the greatest integer less than or equal to x. E.g. $[3.5] = 3$, $[-1.5] = -2$.

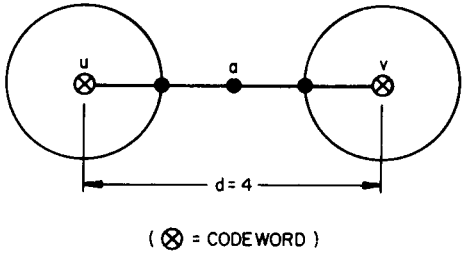

Fig. 1.5. A code with minimum distance 4.

On the other hand, if more than $d/2$ errors occur, the received vector may or may not be closer to some other codeword than to the correct one. If it is, the decoder will be fooled and will output the wrong codeword. This is called a *decoding error.* Of course with a good code this should rarely happen.

So far we have assumed that the decoder will *always* try to find the nearest codeword. This scheme, called *complete decoding*, is fine for messages which can't be retransmitted, such as a photograph from Mars, or an old magnetic tape. In such a case we want to extract as much as possible from the received vector.

But often we want to be more cautious, or cannot afford the most expensive decoding method. In such cases we might use an *incomplete* decoding strategy: if it appears that no more than l errors occurred, correct them, otherwise reject the message or ask for a retransmission.

Error detection is an extreme version of this, when the receiver makes no attempt to correct errors, but just tests the received vector to see if it is a codeword. If it is not, he *detects* that an error has occurred and asks for a retransmission of the message. This scheme has the advantages that the algorithm for detecting errors is very simple (see the next section) and the probability of an undetected error is very low (§5). The disadvantage is that if the channel is bad, too much time will be spent retransmitting, which is an inefficient use of the channel and produces unpleasant delays.

Nonbinary codes. Almost everything we have said applies equally well to codes over other fields. If the field F has q elements, then the message $\boldsymbol{u}$, the codeword $\boldsymbol{x}$, the received vector $\boldsymbol{y}$, and the error vector

$$\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{x} = e_1 e_2 \cdots e_n$$

all have components from F.

We assume that e_i is 0 with probability $1 - p > \frac{1}{2}$, and e_i is any of the $q - 1$ nonzero elements of F with probability $p/(q-1)$. In other words the channel **is a *q-ary symmetric channel*, with q inputs, q outputs, a probability $1 - p > 1/q$ that no error occurs, and a probability $p < (q-1)/q$ that an error does occur, each of the $q - 1$ possible errors being equally likely.**

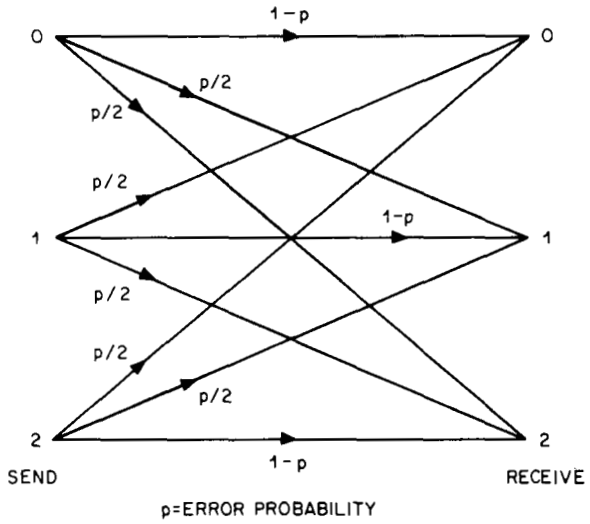

Fig. 1.6. The ternary symmetric channel.

Example. If $q = 3$, the ternary symmetric channel is shown in Fig. 1.6.

Problems. (8) Show that for binary vectors:

(a) $$\mathrm{wt}(\boldsymbol{x}+\boldsymbol{y}) \geqslant \mathrm{wt}(\boldsymbol{x}) - \mathrm{wt}(\boldsymbol{y}), \tag{15}$$

with equality iff $x_i = 1$ whenever $y_i = 1$.

(b) $$\mathrm{wt}(\boldsymbol{x}+\boldsymbol{z}) + \mathrm{wt}(\boldsymbol{y}+\boldsymbol{z}) + \mathrm{wt}(\boldsymbol{x}+\boldsymbol{y}+\boldsymbol{z}) \geqslant 2\,\mathrm{wt}(\boldsymbol{x}+\boldsymbol{y}+\boldsymbol{x}*\boldsymbol{y}) - \mathrm{wt}(\boldsymbol{z}), \tag{16}$$

with equality iff it never happens that x_i and y_i are 0 and z_i is 1.

(9) Define the *product* of vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ from any field to be the vector

$$\boldsymbol{x} * \boldsymbol{y} = (x_1 y_1, \ldots, x_n y_n).$$

For binary vectors this is called the intersection – see Problem 7. Show that for ternary vectors,

$$\mathrm{wt}(\boldsymbol{x}+\boldsymbol{y}) = \mathrm{wt}(\boldsymbol{x}) + \mathrm{wt}(\boldsymbol{y}) - f(\boldsymbol{x}*\boldsymbol{y}), \tag{17}$$

where, if $\boldsymbol{u} = u_1 \cdots u_n$ is a ternary vector containing a 0's, b 1's and c 2's, then $f(\boldsymbol{u}) = b + 2c$. Therefore

$$\mathrm{wt}(\boldsymbol{x}) + \mathrm{wt}(\boldsymbol{y}) - 2\,\mathrm{wt}(\boldsymbol{x}*\boldsymbol{y}) \leqslant \mathrm{wt}(\boldsymbol{x}+\boldsymbol{y}) \leqslant \mathrm{wt}(\boldsymbol{x}) + \mathrm{wt}(\boldsymbol{y}) - \mathrm{wt}(\boldsymbol{x}*\boldsymbol{y}). \tag{18}$$

(10) Show that in a linear binary code, either all the codewords have even weight, or exactly half have even weight and half have odd weight.

(11) Show that in a linear binary code, either all the codewords begin with 0, or exactly half begin with 0 and half with 1. Generalize!

Problems 10 and 11 both follow from:

(12) Suppose G is an abelian group which contains a subset A with three properties: (i) if $a_1, a_2 \in A$, then $a_1 - a_2 \in A$, (ii) if $b_1, b_2 \notin A$, then $b_1 - b_2 \in A$, (iii) if $a \in A$, $b \notin A$, then $a + b \notin A$. Show that either $A = G$, or A is a subgroup of G and there is an element $b \notin A$ such that

$$G = A \cup (b + A),$$

where $b + A$ is the set of all elements $b + a$, $a \in A$.

(13) Show that Hamming distance obeys the *triangle inequality*: for any vectors $\boldsymbol{x} = x_1 \cdots x_n$, $\boldsymbol{y} = y_1 \cdots y_n$, $\boldsymbol{z} = z_1 \cdots z_n$,

$$\text{dist}(\boldsymbol{x}, \boldsymbol{y}) + \text{dist}(\boldsymbol{y}, \boldsymbol{z}) \geqslant \text{dist}(\boldsymbol{x}, \boldsymbol{z}). \tag{19}$$

Show that equality holds iff, for all i, either $x_i = y_i$ or $y_i = z_i$ (or both).

(14) Hence show that if a code has minimum distance d, the codeword $\boldsymbol{x}$ is transmitted, not more than $\frac{1}{2}(d-1)$ errors occur, and $\boldsymbol{y}$ is received, then $\text{dist}(\boldsymbol{x}, \boldsymbol{y}) < \text{dist}(\boldsymbol{y}, \boldsymbol{z})$ for all codewords $\boldsymbol{z} \neq \boldsymbol{x}$. (A more formal proof of Theorem 2.)

(15) Show that a code can simultaneously correct $\leqslant a$ errors and detect $a+1, \ldots, b$ errors iff it has minimum distance at least $a + b + 1$.

(16) Show that if $\boldsymbol{x}$ and $\boldsymbol{y}$ are binary vectors, the Euclidean distance between the points $\boldsymbol{x}$ and $\boldsymbol{y}$ is

$$\left(\sum_{i=1}^{n} (x_i - y_i)^2\right)^{1/2} = \sqrt{(\text{dist}(\boldsymbol{x}, \boldsymbol{y}))}.$$

(17) *Combining two codes* (*I*). Let G_1, G_2 be generator matrices for $[n_1, k, d_1]$ and $[n_2, k, d_2]$ codes respectively. Show that the codes with generator matrices

$$\begin{pmatrix} G_1 & 0 \\ 0 & G_2 \end{pmatrix}$$

and $(G_1|G_2)$ are $[n_1 + n_2, 2k, \min\{d_1, d_2\}]$ and $[n_1 + n_2, k, d]$ codes, respectively, where $d \geqslant d_1 + d_2$.

(18) *Binomial coefficients.* The *binomial coefficient* $\binom{x}{m}$, pronounced "x choose m," is defined by

$$\binom{x}{m} = \begin{cases} \dfrac{x(x-1)\cdots(x-m+1)}{m!}, & \text{if } m \text{ is a positive integer,} \\ 1, & \text{if } m = 0, \\ 0, & \text{otherwise,} \end{cases}$$

where x is any real number, and $m! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (m-1)m$, $0! = 1$. The

reader should know the following properties:

(a) If $x = n$ is a nonnegative integer and $n \geqslant m \geqslant 0$,

$$\binom{n}{m} = \frac{n!}{m!(n-m)!},$$

which is the number of unordered selections of m objects from a set of n objects.

(b) There are $\binom{n}{m}$ binary vectors of length n and weight m. There are $(q-1)^m\binom{n}{m}$ vectors from a field of q elements which have length n and weight m. [For each of the m nonzero coordinates can be filled with any field element except zero.]

(c) The binomial series:

$$(a+b)^n = \sum_{m=0}^{n} \binom{n}{m} a^{n-m}b^m, \quad \text{if } n \text{ is a nonnegative integer;}$$

$$(1+b)^x = \sum_{m=0}^{\infty} \binom{x}{m} b^m, \quad \text{for } |b|<1 \text{ and any real } x;$$

$$\frac{1}{(1-b)^{x+1}} = \sum_{r=0}^{\infty} \binom{x+r}{r} b^r, \quad \text{for } |b|<1 \text{ and any real } x.$$

[Remember the student who, when asked to expand $(a+b)^n$ on an exam replied: $(a+b)^n$, $(a+b)^n$, $(a+b)^n$, $(a+b)^n$, $(a+b)^n$, ...?]

(d) *Easy identities.* (Here n and m are nonnegative integers, x is any real number)

$$\binom{n}{m} = \binom{n}{n-m},$$

$$\binom{n}{m} = 0 \quad \text{for } m > n \geqslant 0,$$

$$\binom{n}{m} + \binom{n}{m-1} = \binom{n+1}{m},$$

$$(-1)^m\binom{-x}{m} = \binom{x+m-1}{m},$$

$$\sum_{m=0}^{n} \binom{n}{m} = 2^n,$$

$$\sum_{m \text{ even}} \binom{n}{m} = \sum_{m \text{ odd}} \binom{n}{m} = 2^{n-1} \quad \text{if } n \geqslant 1,$$

$$\sum_{m=0}^{n} (-1)^m\binom{n}{m} = 0 \quad \text{if } n \geqslant 1,$$

$$\sum_{m \text{ divisible by } 4} \binom{n}{m} = \tfrac{1}{4}[2^n + (1+i)^n + (1-i)^n],\ i = \sqrt{-1}. \tag{20}$$

(19) Suppose $\boldsymbol{u}$ and $\boldsymbol{v}$ are binary vectors with $\operatorname{dist}(\boldsymbol{u}, \boldsymbol{v}) = d$. Show that the number of vectors $\boldsymbol{w}$ such that $\operatorname{dist}(\boldsymbol{u}, \boldsymbol{w}) = r$ and $\operatorname{dist}(\boldsymbol{v}, \boldsymbol{w}) = s$ is $\binom{d}{i}\binom{n-d}{r-i}$,

where $i = (d + r - s)/2$. If $d + r - s$ is odd this number is 0, while if $r + s = d$ it is $\binom{d}{r}$.

(20) Let $\boldsymbol{u}$, $\boldsymbol{v}$, $\boldsymbol{w}$ and $\boldsymbol{x}$ be four vectors which are pairwise distance d apart; d is necessarily even. Show that there is exactly one vector which is at a distance of $d/2$ from each of $\boldsymbol{u}$, $\boldsymbol{v}$ and $\boldsymbol{w}$. Show that there is at most one vector at distance $d/2$ from all of $\boldsymbol{u}$, $\boldsymbol{v}$, $\boldsymbol{w}$ and $\boldsymbol{x}$.

§4. More about decoding a linear code

Definition of a coset. Let $\mathscr{C}$ be an $[n, k]$ linear code over a field with q elements. For any vector $\boldsymbol{a}$, the set

$$\boldsymbol{a} + \mathscr{C} = \{\boldsymbol{a} + \boldsymbol{x} : \boldsymbol{x} \in \mathscr{C}\}$$

is called a *coset* (or *translate*) of $\mathscr{C}$. Every vector $\boldsymbol{b}$ is in some coset (in $\boldsymbol{b} + \mathscr{C}$ for example). $\boldsymbol{a}$ and $\boldsymbol{b}$ are in the same coset iff $(\boldsymbol{a} - \boldsymbol{b}) \in \mathscr{C}$. Each coset contains q^k vectors.

Proposition 3. *Two cosets are either disjoint or coincide (partial overlap is impossible).*

Proof. If cosets $\boldsymbol{a} + \mathscr{C}$ and $\boldsymbol{b} + \mathscr{C}$ overlap, take $\boldsymbol{v} \in (\boldsymbol{a} + \mathscr{C}) \cap (\boldsymbol{b} + \mathscr{C})$. Then $\boldsymbol{v} = \boldsymbol{a} + \boldsymbol{x} = \boldsymbol{b} + \boldsymbol{y}$, where $\boldsymbol{x}$ and $\boldsymbol{y} \in \mathscr{C}$. Therefore $\boldsymbol{b} = \boldsymbol{a} + \boldsymbol{x} - \boldsymbol{y} = \boldsymbol{a} + \boldsymbol{x}'$ $(\boldsymbol{x}' \in \mathscr{C})$, and so $\boldsymbol{b} + \mathscr{C} \subset \boldsymbol{a} + \mathscr{C}$. Similarly $\boldsymbol{a} + \mathscr{C} \subset \boldsymbol{b} + \mathscr{C}$, and so $\boldsymbol{a} + \mathscr{C} = \boldsymbol{b} + \mathscr{C}$. Q.E.D.

Therefore the set F^n of all vectors can be partitioned into cosets of $\mathscr{C}$:

$$F^n = \mathscr{C} \cup (\boldsymbol{a}_1 + \mathscr{C}) \cup (\boldsymbol{a}_2 + \mathscr{C}) \cup \cdots \cup (\boldsymbol{a}_t + \mathscr{C}) \tag{21}$$

where $t = q^{n-k} - 1$.

Suppose the decoder receives the vector $\boldsymbol{y}$. $\boldsymbol{y}$ must belong to some coset in (21), say $\boldsymbol{y} = \boldsymbol{a}_i + \boldsymbol{x}$ $(\boldsymbol{x} \in \mathscr{C})$. What are the possible error vectors $\boldsymbol{e}$ which could have occurred? If the codeword $\boldsymbol{x}'$ was transmitted, the error vector is $\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{x}' = \boldsymbol{a}_i + \boldsymbol{x} - \boldsymbol{x}' = \boldsymbol{a}_i + \boldsymbol{x}'' \in \boldsymbol{a}_i + \mathscr{C}$. We deduce that:

> the possible error vectors are exactly the vectors in the coset containing $\boldsymbol{y}$.

So the decoder's strategy is, given $\boldsymbol{y}$, to choose a minimum weight vector $\hat{\boldsymbol{e}}$ in the coset containing $\boldsymbol{y}$, and to decode $\boldsymbol{y}$ as $\hat{\boldsymbol{x}} = \boldsymbol{y} - \hat{\boldsymbol{e}}$. The minimum weight vector in a coset is called the *coset leader.* (If there is more than one vector with the minimum weight, choose one at random and call it the coset leader.)

We assume that the $\boldsymbol{a}_i$'s in (21) are the coset leaders.

The standard array. A useful way to describe what the decoder does is by a table, called a *standard array* for the code. The first row consists of the code itself, with the zero codeword on the left:

$$\boldsymbol{x}^{(1)} = \mathbf{0}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(s)}, \qquad (s = q^k);$$

and the other rows are the other cosets $\boldsymbol{a}_i + \mathscr{C}$, arranged in the same order and with the coset leader on the left:

$$\boldsymbol{a}_i + \boldsymbol{x}^{(1)}, \qquad \boldsymbol{a}_i + \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{a}_i + \boldsymbol{x}^{(s)}.$$

Example. *Code # 4* (cont.). The [4, 2] code with generator matrix

$$G = \begin{bmatrix} 1011 \\ 0101 \end{bmatrix}$$

has a standard array shown in Fig. 1.7 (ignore the last column for the moment).

message:	00	10	01	11	syndrome $\boldsymbol{S}$
code:	0000	1011	0101	1110	$\binom{0}{0}$
coset:	1000	0011	1101	0110	$\binom{1}{1}$
coset:	0100	1111	0001	1010	$\binom{0}{1}$
coset:	0010	1001	0111	1100	$\binom{1}{0}$
	coset leaders				

Fig. 1.7. *A standard array.*

Note that all 16 vectors of length 4 appear, divided into the 4 cosets forming the rows, and the coset leaders are on the left.

Here is how the decoder uses the standard array: When $\boldsymbol{y}$ is received (e.g. 1111) its position in the array is found. Then the decoder decides that the error vector $\hat{\boldsymbol{e}}$ is the coset leader found at the extreme left of $\boldsymbol{y}$ (0100), and $\boldsymbol{y}$ is decoded as the codeword $\hat{\boldsymbol{x}} = \boldsymbol{y} - \hat{\boldsymbol{e}} = 1011$ found at the top of the column containing $\boldsymbol{y}$. (The corresponding message is 10.)

Decoding using a standard array is maximum likelihood decoding.

The syndrome. There is an easy way to find which coset $\boldsymbol{y}$ is in: compute the vector

$$\boldsymbol{S} = H\boldsymbol{y}^{\mathrm{tr}},$$

which is called the *syndrome* of $\boldsymbol{y}$.

Properties of the syndrome. (1) $\boldsymbol{S}$ is a column vector of length $n - k$.

(2) The syndrome of $\boldsymbol{y}$, $\boldsymbol{S} = H\boldsymbol{y}^{\mathrm{tr}}$, is zero iff $\boldsymbol{y}$ is a codeword (by definition of the code). So if no errors occur, the syndrome of $\boldsymbol{y}$ is zero (but not

conversely). In general, if $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{e}$ where $\boldsymbol{x} \in \mathscr{C}$, then

$$\boldsymbol{S} = H\boldsymbol{y}^{\mathrm{tr}} = H\boldsymbol{x}^{\mathrm{tr}} + H\boldsymbol{e}^{\mathrm{tr}} = H\boldsymbol{e}^{\mathrm{tr}}. \tag{22}$$

(3) For a binary code, if there are errors at locations $a, b, c, \ldots$, so that

$$\boldsymbol{e} = 0 \cdots 0 \underset{a}{1} 0 \cdots \underset{b}{1} \cdots \underset{c}{1} \cdots 0,$$

then from Equation (22),

$$\boldsymbol{S} = \sum_i e_i \boldsymbol{H}_i \quad (\boldsymbol{H}_i = i^{\text{th}} \text{ column of } H)$$

$$= \boldsymbol{H}_a + \boldsymbol{H}_b + \boldsymbol{H}_c + \cdots.$$

In words:

Theorem 4. *For a binary code, the syndrome is equal to the sum of the columns of H where the errors occurred.* [Thus $\boldsymbol{S}$ is called the "syndrome" because it gives the symptoms of the errors.]

(4) Two vectors are in the same coset of $\mathscr{C}$ iff they have the same syndrome. For $\boldsymbol{u}$ and $\boldsymbol{v}$ are in the same coset iff $(\boldsymbol{u} - \boldsymbol{v}) \in \mathscr{C}$ iff $H(\boldsymbol{u} - \boldsymbol{v})^{\mathrm{tr}} = 0$ iff $H\boldsymbol{u}^{\mathrm{tr}} = H\boldsymbol{v}^{\mathrm{tr}}$. Therefore:

Theorem 5. *There is a* 1-1*-correspondence between syndromes and cosets.*

For example, the cosets in Fig. 1.7 are labeled with their syndromes.

Thus the syndrome contains all the information that the receiver has about the errors.

By Property (2), the pure *error detection* scheme mentioned in the last section just consists of testing if the syndrome is zero. To do this, we recompute the parity check symbols using the received information symbols, and see if they agree with the received parity check symbols. I.e., we re-encode the received information symbols. This only requires a copy of the encoding circuit, which is normally a very simple device compared to the decoder (see Fig. 7.8).

Problems. (21) Construct a standard array for code $\#1$. Use it to decode the vectors 110100 and 111111.

(22) Show that if $\mathscr{C}$ is a binary linear code and $\boldsymbol{a} \notin \mathscr{C}$, then $\mathscr{C} \cup (\boldsymbol{a} + \mathscr{C})$ is also a linear code.