The Theory of Error-Correcting Codes

Part I

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

The Theory of Error-Correcting Codes

Part I

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

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

1977

north-holland publishing company
amsterdam · new york · oxford

LCC Number: 76-41296
ISBN: 0 444 85009 0

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

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.

Contents of Part I

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 10. Reed–Solomon and Justesen codes

Chapter 11. MDS codes

Chapter 12. Alternant, Goppa and other generalized BCH codes

Abbreviated contents of Part II

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 $\boldsymbol{u} = u_1u_2 \ldots u_k$

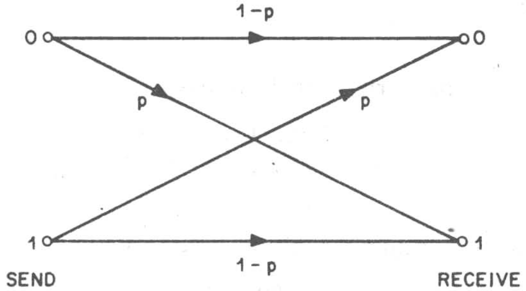

Fig. 1.1. The binary symmetric channel, with error probability p. In general $0 \leqslant p \leqslant \frac{1}{2}$.

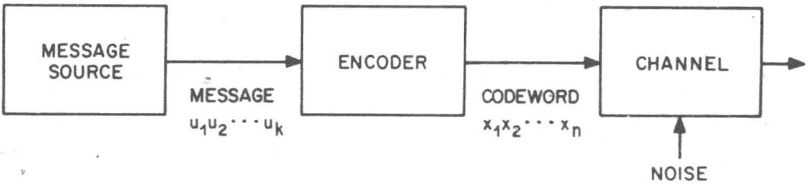

Fig. 1.2.

($u_i = 0$ or 1) will be encoded into a *codeword* $\mathbf{x} = x_1x_2 \dots x_n$ ($x_i = 0$ 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:

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

followed by $n - k$ *check* symbols

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

The check symbols are chosen so that the codewords satisfy

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

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

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

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 $H\boldsymbol{x}^{\text{tr}} = 0$, 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

$$x_4 = -1 - 1 = 1 + 1 = 2 = 0,$$
$$x_5 = -1 = 1, \qquad x_6 = -1 = 1,$$

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 $.00215$ (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

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

(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.