DISCRETE MATHEMATICS AND ITS APPLICATIONS

Introduction to CODING THEORY

Second Edition

Jürgen Bierbrauer

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

Introduction to CODING THEORY

Second Edition

DISCRETE MATHEMATICS AND ITS APPLICATIONS

R. B. J. T. Allenby and Alan Slomson, How to Count: An Introduction to Combinatorics, Third Edition

Craig P. Bauer, Secret History: The Story of Cryptology

Jürgen Bierbrauer, Introduction to Coding Theory, Second Edition

Katalin Bimbó, Combinatory Logic: Pure, Applied and Typed

Katalin Bimbó, Proof Theory: Sequent Calculi and Related Formalisms

Donald Bindner and Martin Erickson, A Student's Guide to the Study, Practice, and Tools of Modern Mathematics

Francine Blanchet-Sadri, Algorithmic Combinatorics on Partial Words

Miklós Bóna, Combinatorics of Permutations, Second Edition

Miklós Bóna, Handbook of Enumerative Combinatorics

Miklós Bóna, Introduction to Enumerative and Analytic Combinatorics, Second Edition

Jason I. Brown, Discrete Structures and Their Interactions

Richard A. Brualdi and Dragoš Cvetković, A Combinatorial Approach to Matrix Theory and Its Applications

Kun-Mao Chao and Bang Ye Wu, Spanning Trees and Optimization Problems

Charalambos A. Charalambides, Enumerative Combinatorics

Gary Chartrand and Ping Zhang, Chromatic Graph Theory

Henri Cohen, Gerhard Frey, et al., Handbook of Elliptic and Hyperelliptic Curve Cryptography

Charles J. Colbourn and Jeffrey H. Dinitz, Handbook of Combinatorial Designs, Second Edition

Abhijit Das, Computational Number Theory

Matthias Dehmer and Frank Emmert-Streib, Quantitative Graph Theory: Mathematical Foundations and Applications

Martin Erickson, Pearls of Discrete Mathematics

Martin Erickson and Anthony Vazzana, Introduction to Number Theory

Titles (continued)

Steven Furino, Ying Miao, and Jianxing Yin, Frames and Resolvable Designs: Uses, Constructions, and Existence

Mark S. Gockenbach, Finite-Dimensional Linear Algebra

Randy Goldberg and Lance Riek, A Practical Handbook of Speech Coders

Jacob E. Goodman and Joseph O'Rourke, Handbook of Discrete and Computational Geometry, Second Edition

Jonathan L. Gross, Combinatorial Methods with Computer Applications

Jonathan L. Gross and Jay Yellen, Graph Theory and Its Applications, Second Edition

Jonathan L. Gross, Jay Yellen, and Ping Zhang Handbook of Graph Theory, Second Edition

David S. Gunderson, Handbook of Mathematical Induction: Theory and Applications

Richard Hammack, Wilfried Imrich, and Sandi Klavžar, Handbook of Product Graphs, Second Edition

Darrel R. Hankerson, Greg A. Harris, and Peter D. Johnson, Introduction to Information Theory and Data Compression, Second Edition

Darel W. Hardy, Fred Richman, and Carol L. Walker, Applied Algebra: Codes, Ciphers, and Discrete Algorithms, Second Edition

Daryl D. Harms, Miroslav Kraetzl, Charles J. Colbourn, and John S. Devitt, Network Reliability: Experiments with a Symbolic Algebra Environment

Silvia Heubach and Toufik Mansour, Combinatorics of Compositions and Words

Leslie Hogben, Handbook of Linear Algebra, Second Edition

Derek F. Holt with Bettina Eick and Eamonn A. O'Brien, Handbook of Computational Group Theory

David M. Jackson and Terry I. Visentin, An Atlas of Smaller Maps in Orientable and Nonorientable Surfaces

Richard E. Klima, Neil P. Sigmon, and Ernest L. Stitzinger, Applications of Abstract Algebra with Maple™ and MATLAB®, Second Edition

Richard E. Klima and Neil P. Sigmon, Cryptology: Classical and Modern with Maplets

Patrick Knupp and Kambiz Salari, Verification of Computer Codes in Computational Science and Engineering

William L. Kocay and Donald L. Kreher, Graphs, Algorithms, and Optimization, Second Edition

Donald L. Kreher and Douglas R. Stinson, Combinatorial Algorithms: Generation Enumeration and Search

Hang T. Lau, A Java Library of Graph Algorithms and Optimization

C. C. Lindner and C. A. Rodger, Design Theory, Second Edition

San Ling, Huaxiong Wang, and Chaoping Xing, Algebraic Curves in Cryptography

Nicholas A. Loehr, Bijective Combinatorics

Toufik Mansour, Combinatorics of Set Partitions

Titles (continued)

Toufik Mansour and Matthias Schork, Commutation Relations, Normal Ordering, and Stirling Numbers

Alasdair McAndrew, Introduction to Cryptography with Open-Source Software

Elliott Mendelson, Introduction to Mathematical Logic, Fifth Edition

Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone, Handbook of Applied Cryptography

Stig F. Mjølsnes, A Multidisciplinary Introduction to Information Security

Jason J. Molitierno, Applications of Combinatorial Matrix Theory to Laplacian Matrices of Graphs

Richard A. Mollin, Advanced Number Theory with Applications

Richard A. Mollin, Algebraic Number Theory, Second Edition

Richard A. Mollin, Codes: The Guide to Secrecy from Ancient to Modern Times

Richard A. Mollin, Fundamental Number Theory with Applications, Second Edition

Richard A. Mollin, An Introduction to Cryptography, Second Edition

Richard A. Mollin, Quadratics

Richard A. Mollin, RSA and Public-Key Cryptography

Carlos J. Moreno and Samuel S. Wagstaff, Jr., Sums of Squares of Integers

Gary L. Mullen and Daniel Panario, Handbook of Finite Fields

Goutam Paul and Subhamoy Maitra, RC4 Stream Cipher and Its Variants

Dingyi Pei, Authentication Codes and Combinatorial Designs

Kenneth H. Rosen, Handbook of Discrete and Combinatorial Mathematics

Yongtang Shi, Matthias Dehmer, Xueliang Li, and Ivan Gutman, Graph Polynomials

Douglas R. Shier and K.T. Wallenius, Applied Mathematical Modeling: A Multidisciplinary Approach

Alexander Stanoyevitch, Introduction to Cryptography with Mathematical Foundations and Computer Implementations

Jörn Steuding, Diophantine Analysis

Douglas R. Stinson, Cryptography: Theory and Practice, Third Edition

Roberto Tamassia, Handbook of Graph Drawing and Visualization

Roberto Togneri and Christopher J. deSilva, Fundamentals of Information Theory and Coding Design

W. D. Wallis, Introduction to Combinatorial Designs, Second Edition

W. D. Wallis and J. C. George, Introduction to Combinatorics

Jiacun Wang, Handbook of Finite State Based Models and Applications

Lawrence C. Washington, Elliptic Curves: Number Theory and Cryptography, Second Edition

DISCRETE MATHEMATICS AND ITS APPLICATIONS

Introduction to CODING THEORY

Second Edition

Jürgen Bierbrauer

Michigan Technological University
Houghton, USA

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

A CHAPMAN & HALL BOOK

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed on acid-free paper
Version Date: 20160414

International Standard Book Number-13: 978-1-4822-9980-9 (Hardback)

Library of Congress Cataloging-in-Publication Data

Names: Bierbrauer, Juergen.
Title: Introduction to coding theory / Jürgen Bierbrauer.
Description: Second edition. | Boca Raton : Taylor & Francis, 2017. | Series: Discrete mathematics and its applications | "A CRC title." | Includes bibliographical references and index.
Identifiers: LCCN 2016017212 | ISBN 9781482299809 (alk. paper)
Subjects: LCSH: Coding theory.
Classification: LCC QA268 .B48 2017 | DDC 003/.54--dc23
LC record available at https://lccn.loc.gov/2016017212

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

To Stella, Daniel and my mother

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Contents

Preface

The theory of error-correcting codes is a branch of discrete mathematics with close ties to other mathematical disciplines, like design theory, combinatorial theory, linear algebra, finite fields, rings, Galois geometry, geometric algebra, algebraic curves over finite fields and group theory. The best known application is in the transmission of messages over noisy communication channels. Other fields of application are to be found in statistics (design of experiments), cryptography (authentication, the design of ciphers) and in many areas of theoretical computer science.

In this textbook we present a self-contained introduction to mathematical coding theory and to its major areas of application. High school algebra and some exposition to basic linear algebra are sufficient as mathematical background. Part I is designed for use in a one semester undergraduate course. A second semester would start with the theory of cyclic codes. In Part II the emphasis is on cyclic codes, applications of codes, liear programming bounds and the geometric description of linear codes. The mathematical tools are developed along the way. Part III offers a brief introduction to some of the basics of the theory of function fields in one variable (algebraic curves) over a finite field of constants, a basic construction of codes (algebraic-geometric codes) and the properties of some interesting families of examples.

A brief overview

The historical origins of coding theory are in the problem of reliable communication over noisy channels. This is a typical problem of the discipline now called **Information Theory.** Both disciplines, **Coding Theory** and **Information Theory,** originated with Claude Shannon's famous 1948 paper [183]. It contains the celebrated **channel coding theorem** (see Chapter 9) which states roughly that good long codes are guaranteed to exist, without giving a clue how to construct them. Closely related is the development of **Cryptography**. Its aim is to ensure reliable communication in the presence of ill-willed opponents. These problems are rather different. In the coding theory scenario we have to overcome a technical problem (the shortcomings of a communication channel), whereas in Cryptography we have to beat an opponent. Nonetheless the mathematical tools used in these two areas have a large

intersection. Historically the development of both disciplines was boosted by the efforts of World War II. Another famous paper of Claude Shannon, [184] from 1949, is perceived as the origin of modern cryptography.

The information-theoretic problem prompted the definition of a mathematical structure called **error-correcting code** or simply **code.** Coding theory can be seen as the study of error-correcting codes, their construction, bounds on their parameters, their implementation and so forth. The most important parameter is the **minimum distance.** It measures the code's capability of correcting transmission errors.

Progress in coding theory was slow but steady. One important development was the theory of **cyclic codes,** which is traditionally couched in the language of ring theory. Cyclic codes are particularly useful because they admit a fast decoding algorithm. The theory of cyclic codes is a core topic of Part II. It is developed in Chapter 13, preceded by an introduction to some relevant features of finite fields in Chapter 12. Our approach is different from the traditional approach. It is based on the trace and the action of the Galois group. Ring theory does not come into play at all.

Only the single most famous cyclic code, the binary Golay code, is introduced in Part I, along with a closely related structure, the large Witt design (Chapter 7).

The ties between coding theory and several areas of pure mathematics have grown stronger all the time. The most important insight goes back to the early 1980s. It is the discovery, by Goppa and Manin [97, 138], of a close relationship between codes and **algebraic curves** (in algebraic language **function fields**). Algebraic curves are objects of **number theory** and **algebraic geometry,** mainstream mathematical disciplines with a long and rich history. The observation by Goppa and Manin makes it possible to use these number-theoretic tools for the construction of codes. The theory of those **algebraic-geometric codes** (AG-codes) is the objective of Part III. In fact we develop only some of the basics of the theory of algebraic curves with finite fields of constants, just enough to understand the basic construction of algebraic-geometric codes and to study some interesting families of examples.

Coding theory and **combinatorics** are closely connected. As an example, **block designs** are important objects of modern discrete mathematics. For more information see the **CRC Handbook of Combinatorial Designs** [106]. We will encounter them repeatedly in the text. A formal definition is in Chapter 7, where we also derive the large Witt design from the binary Golay code. Other examples of block designs in the text include projective planes, projective and affine geometry over finite fields (Chapter 17), the small Witt design, which is derived from the ternary Golay code in Section 17.1, and the Denniston arcs in the same section.

Linear codes can be studied from a geometric point of view. From this perspective coding theory can be seen as part of **Galois geometry.** The basic objects of Galois geometry are **affine and projective spaces** defined over finite fields (see Hirschfeld [113] or the beginning of Chapter 17). Linear

codes can be equivalently described as sets of points in projective spaces. In many cases the geometric language is more appropriate than the algebraic approach. In Part I we study 3-dimensional codes from this point of view (Chapter 10). This case is particularly easy to understand as the underlying geometrical structures are classical projective planes $PG(2, q)$. The general mechanism is developed and used in Chapter 17. In many cases this leads to a better understanding of the codes. For instance, we use the geometric method to construct the ternary Golay code in Section 17.1. As a natural generalization, the additive codes (network codes) of Chapter 18 are described geometrically by families of subspaces of a fixed projective space. As a special case, the binary quantum codes of Chapter 18 are described by families of lines in binary projective spaces.

Caps are sets of points in projective or affine geometry no three of which are on a line. They are formally equivalent to linear codes of minimum distance $d = 4$. It turns out that caps are best understood from a geometric point of view. This is why we study caps in Chapter 17. The case of caps in projective planes and 3-spaces leads to another link with classical algebra. In fact, parabolic and elliptic quadrics yield canonical examples of caps in those dimensions. We include a self-contained introduction to **geometric algebra** in Section 17.2 which gives a better understanding of those caps.

Duality is emphasized throughout the text. The dual of a linear code with minimum distance d is an **orthogonal array** of **strength** $d - 1$. Originally orthogonal arrays were defined in the framework of design of experiments, in **statistics.** The same is true of block designs. The defining properties of orthogonal arrays and of block designs are both uniformity conditions. They look very similar. Orthogonal arrays can be interpreted as families of **random variables** (functions defined on sample spaces), which satisfy certain statistical independence conditions (see Chapter 6). The strength measures the degree of statistical independence. Such families of random variables are heavily used not only in statistics but also in the theory of **algorithms.** Whenever we construct a good linear code, we also obtain such a statistical object.

Typically in coding theory duality is defined with respect to the usual dot product, the Euclidean bilinear form. However, each non-degenerate **bilinear** (or sesquilinear) **form** defines a notion of duality. An application to the construction of **quantum codes** in Chapter 18 demands the use of a special bilinear form, the symplectic form. This is another motivation for covering the theory of bilinear forms in Chapter 17.

Some of the classical **bounds,** the Singleton bound, the Hamming bound, the Plotkin bound and the Griesmer bound on codes as well as the Bose-Bush bound on orthogonal arrays of strength 2, are derived when they are needed in the text. In fact, there is a multitude of bounds, each of which is better than all the others in certain parameter ranges, both for codes (when the minimum distance is the central parameter) and for orthogonal arrays (when the strength is in the center of attention). A general algebraic mechanism for

the derivation of bounds is related to **orthogonal polynomials** (in the case of codes these are the **Kravchouk polynomials**) and **linear programming.** There is a general **linear programming bound.** All the bounds used in the text, with the exception of the Griesmer bound, are special cases of the LP-bound. Chapter 15 is a self-contained introduction to linear programming and contains unified proofs for all those explicit bounds on codes and orthogonal arrays. On the level of LP-bounds there is a relation of duality between bounds on codes and bounds on orthogonal arrays. This is another reason why the notion of an orthogonal array should be seen as the dual of the notion of an error-correcting code, even in the nonlinear case.

This leads us to applications of codes. Traditionally coding theorists are biased towards the information-theoretic application that we encounter so often in this text. It still is one of the major applications. We use it as a motivation in the early chapters, discuss syndrome decoding in Chapter 3 and the decoding algorithm of BCH-codes based on the Euclidean algorithm in Chapter 13. There is, however, a plethora of applications of a completely different nature. Many of them have surfaced in theoretical computer science, in particular in cryptography. **Universal hash families** yield a nice paradigmatic example. One version is presented in Part I, Chapter 6, while a more in depth treatment is in Chapter 16. This chapter is dedicated to applications altogether. They range from statistics and cryptography to numerical integration and the theory of algorithms.

The plan: Part I

Part I forms an elementary introduction to the theory of codes and some typical applications. It assumes only high school mathematics. Some exposition to the basics of linear algebra would be helpful as well.

Chapter 1 introduces some basic concepts like bits and bitstrings and describes the basic problem when messages are sent via a noisy channel. The transmission of pictures from space serves as an illustration. The first steps toward the algebraization of the problem are taken by introducing the field $\mathbb{F}_2$ of two elements and giving the bitstrings of length n the structure of a vector space. We encounter the basic idea of error correction and the basic notion of Hamming distance. This leads to the formal definition of a binary code and of a q-ary code for an arbitrary natural number $q \geq 2$. The binary symmetric channel is the most elementary model which describes how errors are introduced into the message. Basic facts on binomial numbers are reviewed. The sphere-packing bound (or Hamming bound) is our first general bound on codes. The football pool problem is a possible application of ternary ($q = 3$) codes.

Chapter 2 introduces the basics of binary **linear** codes. Key notions are minimum weight, dimension, generator matrix. After a review of basic facts of linear algebra (applied for vector spaces = codes over $\mathbb{F}_2$), we see how block coding works and study the effect on the probability of transmission errors. The dual code is defined with respect to the dot product. The repetition codes and the sum 0 codes are our first pair of dual codes. A check matrix of a code is a generator matrix of its dual code. An important family of codes are the binary Hamming codes. Their duals are the binary Simplex codes. The principle of duality shows how to read off the minimum distance from a check matrix. This leads to the notion of a binary orthogonal array (OA). A linear OA has strength t if and only if its dual has minimum weight $t + 1$.

Chapter 3 generalizes linear codes from the binary ($q = 2$) to the q-ary, where q is an arbitrary prime-power. It is shown how finite fields $\mathbb{F}_q$ of q elements can be constructed. The definition of linear q-ary codes is given and the basic facts generalized from the binary to the q-ary: dimension, generator matrix, dual code, check matrix, principle of duality, orthogonal arrays. Basic methods of linear algebra are reviewed (rank, Gauß elimination, determinants). Mutually orthogonal Latin squares are recognized as special parametric cases of OA. The MacWilliams formula links the weight distribution of a linear code and its dual. Our proof is probabilistic. This motivates the definition of probability spaces. The game of SET leads to natural questions concerning ternary ($q = 3$) linear codes. Syndrome decoding can in principle be used to decode linear codes.

A large and important family of linear codes are the Reed-Solomon codes of Chapter 4. They meet the Singleton bound with equality (they are MDS-codes). Lagrange interpolation shows that they form OA of index $\lambda = 1$. The dual of an RS-code is an RS-code. Mutually orthogonal Latin squares yield non-linear MDS-codes. Covering arrays and their use in software testing are discussed.

Chapter 5 introduces some recursive constructions of codes: shortening and puncturing, the parity check bit for binary codes, the residual code and the Griesmer bound, concatenation and the $(u, u + v)$-construction.

Chapter 6 presents our first application in computer science. The concept of universal hashing is introduced. ϵ-universal hash classes turn out to be formally equivalent with codes.

Chapter 7 is a direct construction of the binary Golay code and the large Witt design. This motivates the definition of t-designs. Classical projective planes $PG(2, q)$ are introduced.

Chapter 8: Meaning and basic properties of the Shannon entropy of a probability space. The binary entropy function. The Jensen inequality as a tool.

In Chapter 9 the concept of asymptotic bounds for infinite families of codes with length $\longrightarrow \infty$ is introduced and the asymptotic version of the Singleton bound given. The Plotkin bound on codes of large distance is proved. It implies an asymptotic bound and the Bose-Bush bound on OA of strength

2. Shannon's channel coding theorem is proved. Basic notions and facts on probability theory (random variable, expectation, variance, Tschebyscheff inequality) are introduced. These are used in the proof. The Gilbert-Varshamov bound is an existence bound on linear codes (certain codes are guaranteed to exist), based on counting arguments. The Justesen codes form an explicit family of codes with asymptotically nontrivial parameters.

Chapter 10 is an introduction to the geometric method. The 3-dimensional codes are described as multisets of points in the projective plane $PG(2, q)$. An application allows the construction of congestion-free networks.

The plan: Part II

Chapter 12 starts with more basic properties of finite fields: Primitive elements, field extensions, the Frobenius automorphism, the Galois group, the trace from a field to a subfield. Trace codes and subfield codes are defined. Delsarte's theorem describes the dual code. We introduce the Galois closure of a linear code with respect to a subfield, prove the second main theorem and sketch the general strategy to construct cyclic codes. Different notions of equivalence of codes are discussed.

Chapter 13 is dedicated to the general machinery of cyclic codes. An example (binary, length 15) is used to illustrate this. This is a subfield code (and a trace code) of a Reed-Solomon code defined over $\mathbb{F}_{16}$. Basic notions of the general construction are cyclotomic cosets, the dimension formula and the BCH-bound on the minimum distance. Parametric examples of cyclic codes are given, as well as an application to fingerprinting. The Roos bound and the van Lint-Wilson method allow improvements on the BCH-bound in special situations. Generator matrices and check matrices of cyclic codes are almost canonically determined. BCH-codes are special cyclic codes. Their decoding algorithm is based on the Euclidean algorithm. Constacyclic codes are generalizations of cyclic codes. They can, however, be described within the theory of cyclic codes. This central chapter ends with two families of particularly good quaternary ($q = 4$) constacyclic codes and a comparison with the traditional ring-theoretic approach to cyclic codes.

Chapter 14 complements Chapter 5 by introducing further recursive constructions of codes. Constructions X and XX can be applied using cyclic codes as ingredients. The covering radius is another basic parameter. It is related to lengthening and has its own applications. We describe an application in steganography and an application in the reduction of switching noise.

Chapter 15 is dedicated to linear programming (LP). The first section is a self-contained introduction to the basics of linear programming. This includes basic notions and results like the simplex algorithm, the Farkas alterna-

tive, duality theorems and the principle of complementary slackness. A basic notion, the **Fourier transform,** is introduced in the second section. This section also contains basic properties of the Kravchouk polynomials and the general LP-bound for codes and orthogonal arrays. Several explicit bounds are derived from the LP-bound in the third section. The fourth and last section of Chapter 15 contains a proof of the celebrated bound of four (McEliece, Rodemich, Rumsey, Welch [140]), an asymptotic bound on codes and orthogonal arrays.

Chapter 16 discusses various applications. OA are interpreted as families of random variables with certain independence properties and as perfect local randomizers. We introduce linear shift register sequences and their relation to Simplex codes, describe the role of minimum distance and strength in the construction of block ciphers and the use of OA in two-point based sampling and chips testing. Further topics include the relation between OA and resilient functions, applications of resilient functions for the wire-tap channel and the generation of random bits, applications of OA in the derandomization of Monte Carlo-algorithms, as well as a more detailed study of universal hash classes, their construction from codes and their applications in cryptography (authentication).

Chapter 17 studies the geometric approach to linear codes. Linear codes are described as multisets of points in projective geometry $PG(k-1,q)$. The main theorem determines the minimum distance in terms of hyperplane intersection sizes. The hexacode, ovals and hyperovals, extended Reed-Solomon codes and the Simplex codes are best understood from this point of view. Codes of dimension 2 and 3 are studied. The ternary Golay code is constructed starting from its parameters $[12,6,6]_3$. Barlotti arcs and Denniston arcs as well as the corresponding codes are described. Caps are sets of points no three of which are on a line. They are formally equivalent to linear codes of minimum distance $d=4$. We introduce the theory of bilinear forms and quadratic forms. This yields canonical models of large caps in $PG(2,q)$ (parabolic quadrics) and in $PG(3,q)$ (elliptic quadrics). Direct constructions of caps in $PG(4,q)$ and general bounds on caps in arbitrary dimension are derived. Recursive constructions of caps use as ingredients elliptic quadrics or, in the ternary case, the Hill cap in $PG(5,3)$.

Chapter 18 introduces codes whose alphabet forms a vector space over a ground field. They generalize the linear codes. Chen projection is a simple recursive construction. Application to caps yields codes which have been used in computer memory systems. Application to Reed-Solomon codes produces codes which have been used in deep space communication. Another recursive construction simplifies the Bose-Bush construction of OA of strength 2. We conclude Section 18.1 with a direct construction of an interesting family of low-dimensional additive codes. A self-contained theory of cyclic additive codes is developed in Section 18.2. It generalizes the approach from Chapter 13. Quaternary additive codes are considered in Section 18.3. Geometrically those are described by multisets of lines in binary projective spaces. We concentrate

on codes of short lengths and obtain the best possible parameters for all lengths ≤ 13. Quantum stabilizer codes are by nature additive (q-linear, q^2-ary) codes which are contained in their symplectic dual. In Section 18.3 we develop the basic theory, using the geometric description in terms of systems of lines and applying the cyclic theory. In particular we determine all parameters of binary ($q = 2$) distance $d = 3$ quantum stabilizer codes.

Additive codes have recently reappeared under the name of network codes (see Koetter and Kschischang [126]). The metric used in [126] is very different from what we used so far. It is based on the ranks of pairwise intersections of the subgeometries describing the code. We describe the solution of the smallest non-trivial problem in Section 18.5. It is surprisingly complicated.

The plan: Part III

In the early chapters of Part III we develop some of the basic theory of function fields of transcendence degree 1 over finite fields of constants, using Stichtenoth's by now classical textbook [198]. Highlights are the Riemann-Roch theorem and the Riemann-Hurwitz formula. As motivating examples we use the Klein quartic, hyperelliptic and Artin-Schreier extensions. The basic construction of codes from algebraic curves (AG-codes) is given in Chapter 22, where we describe some important families of AG-codes and an application to universal hashing. Some additional material is collected in the last chapter. A section on list decoding of Reed-Solomon codes has been included as this represents a rather recent development which uses some basic algebra/geometry in a transparent way. The sections on tms-nets and on sphere packings in Euclidean spaces (Chapter 23) are treated somewhat lighter than the applications in Chapter 16. The theory of tms-nets is to be seen in the context of quasi-Monte Carlo algorithms (related to uniformly distributed point sets in Euclidean space). This application of coding theory to numerical integration and the pricing of exotic options is rather surprising. The construction of dense sphere packings is a classical problem in Euclidean space with links not only to discrete mathematics but also to algebra and algebraic geometry. The remaining sections of Chapter 23 have the character of brief survey articles.

How to use this text

Part I arose from several one semester undergraduate courses on coding theory. Chapters 2 to 6 form the core of such an introductory course. The

only section which can (and maybe should) be skipped is Section 3.5 on the MacWilliams transform. The remaining time should be dedicated to one or several of the later chapters of Part I. Chapter 7 would be a natural choice as the binary Golay code probably is the most famous of all codes. Usually I cover Chapter 6 as it is the first example of a non-standard application and it can be done in one lecture.

Another choice would be to cover Chapter 8 and some of the material from Chapter 9. These chapters form a unit as the entropy function is used in the asymptotic expressions of Chapter 9.

The canonical starting point for the second semester of a two semester course is the theory of cyclic codes and their implementation, Chapters 12 and 13, as well as Chapter 14. Some of the later parts of Chapter 13 are optional (the application to fingerprinting, the Roos bound, the van Lint-Wilson bound, the comparison with the traditional approach and Section 13.4 on constacyclic codes). Section 13.3 on the decoding algorithm of BCH-codes may be sacrificed as well.

Chapter 17 may be considered another core area of coding theory as it gives a better understanding of many important codes. It would be a pity to sacrifice the main theorem of the first section. Some of the applications in the first section ought to be covered.

From here on there are several choices.

For a thorough introduction to the theory of codes and its links with Galois geometry one might concentrate entirely on Chapter 17.

Another possibility is to cut short on Chapter 17 and to cover several applications from Chapter 16 instead. The sections on tms-nets and on sphere packings in Chapter 23 are then a good choice to round off a graduate course. This also has the advantage that the course ends with an introduction to an exceptional object, the Leech lattice.

A third strategy is to concentrate on the theory of cyclic codes and their applications. Such a course would end with Chapter 18.

The database

Tables on parameters for linear and quantum codes are to be found in M. Grassl's page

`http://www.codetables.de`

What has changed in the second edition?

The changes are too numerous to be listed exhaustively. A macroscopic change is the addition of a new Part III in the second edition, dedicated to algebraic-geometric codes. A little section in the last chapter of the first edition has turned into a new Chapter 15 of the second edition, containing an introduction to linear programming and the derivation of explicit bounds on codes and orthogonal arrays. Chapter 18 on additive codes and network codes is a completely remodeled version of the corresponding chapter in the first edition. The cyclic case is based on a more general theory, the geometric approach has been greatly expanded and there is an additional section on network codes. The last chapter has undergone a mutation as well. Two of its sections in the first edition have vanished as they turned into chapters of their own. Sections on permutation codes and on highly symmetric codes have been added as well as two sections which should make the text more readable, one on the individual small fields that have been used in the text and another section on the individual short codes constructed in the text.

Textbooks

An early classic among the textbooks is *Algebraic Coding Theory* [11] by E. R. Berlekamp, which presents in particular an excellent introduction to the traditional theory of cyclic codes. The 1977 book *The Theory of Error-Correcting Codes* [142] by MacWilliams and Sloane is considered something like the bible of traditional coding theory. A more recent introduction is J. H. van Lint's *Introduction to Coding Theory* [211], a relatively short and dense text which helped a great deal in attracting pure mathematicians to the area. H. Stichtenoth's book *Algebraic Function Fields and Codes* [198] is a self-contained introduction to the theory of algebraic function fields in one variable (equivalently: algebraic curves) and to the codes derived from them. Among the undergraduate textbooks we mention Vera Pless, *The Theory of Error-Correcting Codes* [165] and R. Hill, *A first course in coding theory* [112]. Hill's book is the first of its kind presenting an introduction to the geometric approach to codes.

Orthogonal Arrays: Theory and Applications [109] by Hedayat, Sloane and Stufken introduces to orthogonal arrays (dual codes) and their applications to the design of experiments.

Acknowledgments

My thanks go to Yves Edel; Wolfgang Ch. Schmid, Peter Hellekalek and the Institut für Mathematik of the University of Salzburg; Fernanda Pambianco, Stefano Marcugini, Massimo Giulietti, Daniele Bartoli, Giorgio Faina and the dipartimento di matematica e informatica of the University of Perugia; Bernd Stellmacher and the Mathematisches Seminar of the University of Kiel; C. L. Chen, Ron Crandall, Ludo Tolhuizen, Albrecht Brandis, Allan Struthers, Michigan Technological University; Gary Ebert; and the National Security Agency.

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

About the author

Jürgen Bierbrauer received his PhD in mathematics at the University of Mainz (Germany) in 1977, with a dissertation about the theory of finite groups. His mathematical interests are in algebraic methods of discrete mathematics and its applications. He held a position at the University of Heidelberg (Germany) from 1977 to 1994 and the position of full professor at Michigan Technological University in Houghton, Michigan until his retirement in 2015. His non-mathematical interests include Romance languages, literature, and the game of Go.

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Part I

An elementary introduction to coding

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Chapter 1

The concept of coding

1.1 Bitstrings and binary operations

Basic concepts: bits, bitstrings, transmission of messages, transmitting pictures from space, the Morse code, XORing, the field $\mathbb{F}_2$, the model of message transmission, a first idea of error correction.

The object of coding theory is the transmission of messages over noisy channels. Figure 1.1 shows the standard picture visualizing the situation.

At first we need to understand what the elements of this picture mean: what is a **message,** a **channel,** what is **noise**? Along the way we will encounter more basic notions. In this first chapter some of these will be explained. We start with the message.

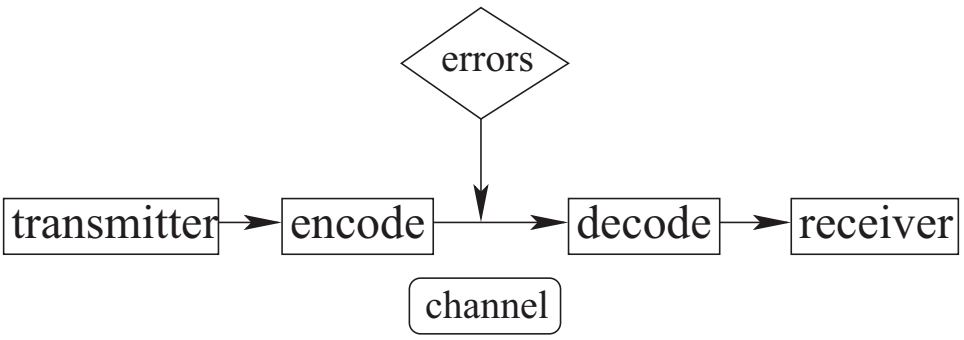

FIGURE 1.1: Information transmission over a noisy channel

If there are 8 possible messages to be sent, say, then we can represent each message as a bitstring of length 3, like 000 or 011 or 110. We will generally assume this has been done and define a **message** to be a bitstring. Here are some examples of bitstrings and their lengths:

bit string	length
000	3
110	3
110011	6
0000011111	10

Transmitting pictures from space

Assume we wish to transmit a photograph from outer space, like one of the pictures of Saturn taken by the Voyager spacecrafts in the early 1980s (*Viger* for Star Trek buffs). The picture is divided into 800×800 pixels; each pixel is assigned one of $256 = 2^8$ degrees of brightness. The brightness of a pixel is thus represented by a bitstring of length 8 and the total black and white picture consists of $800 \times 800 \times 8$ bits. As the picture really is in color, the same photo is transmitted three times, each time through a different color filter. The full color picture will thus be represented by a bitstring of length $3 \times 800 \times 800 \times 8 = 15,360,000$. This is our message. The channel is determined by the properties of space between the spacecraft and the receiver on Earth, above all by the Earth's atmosphere. A certain number of bits will be destroyed. Here we only consider errors of the type that 0 is transmitted and 1 is received or vice versa.

The Morse code

Another illustration for the claim that every message can be represented by bitstrings is the Morse code, which has been in use for telegraphy since the 1840s. It represents each of the 26 letters A, B, ... , Z, each digit $0, 1, \ldots, 9$ as well as the period, the comma and the question mark by a sequence of at most five dots and dashes. The dot stands for a short signal, the dash for a long signal. Dashes are about three times as long as dots. For example, the

letter E is represented by a single dot, S is represented by dot-dot-dot and Z is dash-dash-dot-dot.

However, the graphical representation is purely conventional. We can represent a dash by 1, a dot by 0 and obtain a representation of letters and numbers as bitstrings: E=0, S=000, Z=1100, and T=1, Q=1101, V=0001.

Back to the general model

Assume we wish to send one of 8 possible messages (the bitstrings of length 3), for example, message 011. If it should happen along the way (in the channel) that the second bit is flipped (the second coordinate is in error), then 001 will be received.

We want to express this situation in mathematical terminology.

1.1 Definition. $\mathbb{F}_2 = \{0, 1\}$ *with addition*

$$0 + 0 = 0,\ 0 + 1 = 1,\ 1 + 0 = 1,\ 1 + 1 = 0$$

Here the letter $\mathbb{F}$ stands for **field.** It indicates that this tiny structure of only two elements has something in common with the field of real numbers and the complex number field: each of these structures satisfies the same set of axioms. Each of them is a field. We will come back to this topic later.

The idea behind the addition in $\mathbb{F}_2$ is to model the errors in information transmission: a coordinate of a message is in error if and only if 1 is added to the entry in this coordinate. This explains why we must have $1 + 1 = 0$: if a bit is flipped twice, then no error occurs, or: if 1 is flipped, then 0 is received.

Addition in $\mathbb{F}_2$ is also known as XORing, in particular in the computer science literature. Here **XOR** stands for **exclusive or,** a logical operation, where 1 stands for **true** and 0 for **false.** The relation $1 + 1 = 0$ (true or true = false) is what makes it "exclusive": in order for the result to be true, one of the two ingredients has to be true, but not both (the ordinary **or** operation would have $1 + 1 = 1$).

Another motivation for our binary addition comes from arithmetic: if we distinguish even and odd integers, the following familiar rules hold:

$$even + odd = odd + even = odd,\ even + even = odd + odd = even.$$

With $even = 0$ and $odd = 1$ these are exactly the rules of binary addition.

Addition in $\mathbb{F}_2$ describes what happens in each coordinate. Calculation with bitstrings is formalized as follows:

1.2 Definition. $\mathbb{F}_2^n$ *consists of bitstrings* $x = (x_1, x_2, \ldots, x_n)$ *of length* n, *where* $x_i \in \mathbb{F}_2$. *Addition in* $\mathbb{F}_2^n$ *is coordinatewise.*

For example, let $x = 001 \in \mathbb{F}_2^3$ and $y = 101 \in \mathbb{F}_2^3$. Then $x + y = 100$. With this terminology it is easier to express the relation between messages, errors and received messages. Let $x = 001100$ be the message sent. Assume errors occur in coordinates 2 and 4. We can express this by adding the **error vector** $e = 010100$. The received message will then be the sum

$$y = x + e = 011000.$$

Let us do this the other way around: if $x = 010101$ was sent and $y = 010111$ was received, then an error occurred in coordinate 5. This means $e = 000010$.

Return to the situation where we send one of eight messages (the elements of $\mathbb{F}_2^3$), for example, $x = 011$. No matter what the error vector e is, the received message $y = x + e$ is again one of the legitimate messages. There is no way for the receiver to suspect that an error occurred, let alone to correct it. So how can we hope to correct errors?

Error correction: The first idea

Here is the easiest of all error-correcting systems: encode each 0 of the message to be sent as a block 000 and analogously each 1 by 111. If the original message is $x = 011$, the encoded message is now $x' = 000111111$. The receiver knows what encoding scheme has been used. He will therefore divide the received message in blocks of length 3. If such a block is 000 or 111, then decoding is obvious: $000 \mapsto 0$, $111 \mapsto 1$. Assume a block of the received message is 101. The receiver knows that at least one transmission error must have happened. It is a basic assumption that a small number of errors is more probable than many errors. The decoding will therefore be $101 \mapsto 1$ (by majority decision).

The initial picture begins to make sense now. We have seen that messages can be encoded by the transmitter and decoded by the receiver such that the following holds: if not more than one error occurs during transmission (in the channel), then the error will automatically be corrected. What we have used is known as the **repetition code** of length 3.

1. We learned how to calculate with bitstrings.
2. $\mathbb{F}_2^n$ consists of the bitstrings of length n.
3. The received message is the binary sum of the sent message and the error vector.
4. The repetition code of length 3 corrects one bit error.

Exercises 1.1

1.1.1. *Compute the sum of* 11001 *and* 01110.

1.1.2. *Assume* 000000 *was sent and two errors occurred. List all possible received messages.*

1.1.3. *Let* $x = 1101$ *be the message to be sent. Encode* x *using the repetition code of length* 3.

1.1.4. *Assume the repetition code of length* 3 *is used and* 000110111101 *is received. What is the result of decoding?*

1.1.5. *Why does the Morse code represent letters E and T by strings of length* 1, *whereas letters like Q, V, Z are represented by longer bitstrings?*

1.2 The Hamming distance

Basic concepts: The Hamming distance as a metric, the weight of a bitstring.

1.3 Definition. *Let* $x = (x_1, x_2, \ldots, x_n)$ *and* $y = (y_1, y_2, \ldots, y_n)$ *be bitstrings in* $\mathbb{F}_2^n$. *The* **distance** *(or* **Hamming distance***) between* x *and* y *is*

$$d(x, y) = \textit{number of coordinates } i \textit{ where } x_i \neq y_i.$$

Here are some examples:

$$d(0000, 1111) = 4,\ d(00110, 00101) = 2,\ d(111111, 001100) = 4.$$

Expressed in the context of messages and errors, $d(x, y)$ is the minimum number of errors transforming x into y. In fact, consider the second example above: $x = 00110$ and $y = 00101$ differ in the last two coordinates, $d(x, y) = 2$ and $x + 00011 = y$.

Things get even easier when we use the weight.

1.4 Definition. *The* **weight** $wt(x)$ *of the bitstring* $x \in \mathbb{F}_2^n$ *is the number of nonzero coordinates in* x.

Here are some examples:

$$wt(0000) = 0,\ wt(1111) = 4,\ wt(00110) = 2,\ wt(001101) = 3.$$

The weight of a bitstring is its distance from the all-0 bitstring. If the all-0 bitstring is sent and w errors occur during transmission, then the received message has weight w. If x is sent, e is the error vector and $y = x + e$ is received, then $d(x, y) = wt(e)$.

The Hamming distance is also called the Hamming metric. The general notion of a metric is widely used in mathematics. Here is the definition:

1.5 Definition. *Let* X *be a set. For every pair* $x \in X,\ y \in X$ *let a real number* $d(x, y)$ *be given (the* **distance** *from* x *to* y*). The function* d *is called a* **metric** *if the following are satisfied:*

- $d(x, y) \geq 0$ *for all* x, y.
- $d(y, x) = d(x, y)$ *for all* x, y.
- $d(x, y) = 0$ *if and only if* $x = y$.
- $d(x, z) \leq d(x, y) + d(y, z)$ *for all* x, y, z.

The last requirement is the most important. It is known as the **triangle inequality.** A famous metric is the Euclidean metric in Euclidean space. If, for example, $x = (x_1, x_2)$ and $y = (y_1, y_2)$ are two points in the plane, then their Euclidean distance is $\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$.

1.6 Theorem. *The Hamming distance is a metric on* $\mathbb{F}_2^n$.

Most of the properties of Definition 1.5 are obvious. Only the triangle inequality is a little interesting. This is left as an exercise.

1. The Hamming distance between two bitstrings of the same length is the number of coordinates where they differ.
2. $d(x, y)$ is the number of bit errors needed to transform x into y.
3. The Hamming distance is a metric.
4. The weight is the distance from the all-0 vector.
5. $d(x, y) = wt(x + y)$.
6. If bitstring x is sent and y is received, then $d(x, y) = wt(e)$ is the weight of the error vector.

Exercises 1.2

1.2.1. *Compute* $d(11001, 01110)$ *and* $d(0000, 0110)$.

1.2.2. *Find* $wt(00110)$ *and* $wt(10111)$.

1.2.3. *List all vectors in* $\mathbb{F}_2^6$ *at distance* 3 *from* 111000.

1.2.4. *The alphabet has* 26 *letters. If we want to represent all possible words of length* ≤ 3 *(all letters, pairs of letters and triples of letters) as bitstrings of the same length* n*, what is the smallest number* n *such that this is possible?*

1.2.5. *Prove that the Hamming distance is a metric.*

1.2.6. *Assume* x *is sent and* $y = x + e$ *is received. What can we say about* $d(x, y)$ *and about* $wt(e)$ *if not more than* 3 *errors have occurred?*

1.3 Binary codes

Basic concepts: Length, minimum distance. The idea of error correction.

We saw that no errors can be detected or corrected if all elements of $\mathbb{F}_2^n$ are used as messages. The obvious idea is to use only a certain subset. Such subsets will be called **codes.**

Let us send bitstrings of length 6. Instead of using all elements of $\mathbb{F}_2^6$ as (encoded) messages, we use only the following subset:

000000	001011
100110	101101
010101	011110
110011	111000

Such a family of bitstrings of length 6 is also called a **binary code** of length 6. Its elements are **codewords.** In our example we have 8 codewords.

The most important property of this code is the following: any two different codewords are at distance ≥ 3. We say that 3 is the **minimum distance** of the code. Please check for yourself that this is true. The parameters of this binary code are then recorded as $(6, 8, 3)_2$: we have a binary code (indicated by subscript 2), of length 6, consisting of 8 codewords, with minimum distance of 3.

The idea of error correction

Transmitter and receiver agree on the code to be used. Only codewords will be sent. If only one error occurs in the channel, then the received word will be in a ball of radius 1 around a codeword (in the Hamming metric). Assume the code has been chosen such that any two codewords are at distance at least 3. Then the balls of radius 1 do not overlap: if a bitstring has distance 1 from some codeword, then it has a larger distance from any other codeword. In other words, the receiver will decode any vector at distance ≤ 1 from some codeword as that codeword.

In the picture: the whole ball (or call it a disc) of radius 1 is decoded as the center of the ball, or: the received tuple is decoded as the codeword which it resembles most closely. If not more than one error occurred, then this error will be corrected. Observe that Figure 1.2 serves only as an illustration.

The metric in the Euclidean plane is used to illustrate the situation in a rather different metric, the Hamming metric.

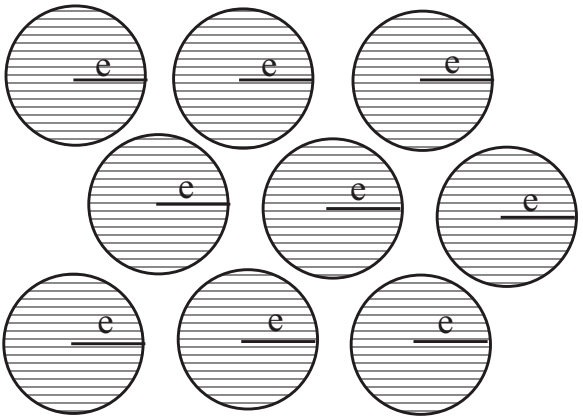

FIGURE 1.2: Non-overlapping balls centered at codewords

1.7 Definition. *A binary code* $(n, M, d)_2$ *is a set of* M *bitstrings in* $\mathbb{F}_2^n$ *such that any two different elements of the code (codewords) are at distance* $\geq d$. *We call* n *the* **length** *and* d *the* **minimum distance** *of the code.*

A code of minimum distance 5 can correct 2 errors, minimum distance 7 can correct 3 errors and so on. A fundamental problem of coding theory is the following: Given n and d, find the largest M such that there is a code $(n, M, d)_2$.

1. A binary code $(n, M, d)_2$ is a collection of M bitstrings of length n such that any two different of these codewords are at Hamming distance at least d.
2. A basic problem of coding theory: determine the maximum M such that an $(n, M, d)_2$-code exists.
3. A code can correct e errors provided its minimum distance is $d \geq 2e + 1$.
4. We saw a $(6, 8, 3)_2-$code.

Exercises 1.3

1.3.1. *If we want to correct* 8 *bit errors, what would the minimum distance of the code have to be?*

1.3.2. *Using our code* $(6, 8, 3)_2$, *decode the following received vectors:*

$$111100,\ 111011,\ 000001,\ 011110.$$

1.3.3. *Does a code* $(5, 6, 3)_2$ *exist?*

1.4 Error-correcting codes in general

Basic concepts: Basic code parameters. Telegraphy codes as early examples.

The notion of a **binary** code is too narrow, although it is most frequently used in information transmission. Here is the general concept of an (error-correcting) code:

1.8 Definition. *Let* $\mathcal{A}$ *be a finite set of* q *elements (the* **alphabet***). A* q*-ary code* $\mathcal{C}$ *of* **length** n *is a family of* n*-tuples with entries in* $\mathcal{A}$ *:*

$$\mathcal{C} \subseteq \mathcal{A}^n.$$

For example, let $q = 3$ and $\mathcal{A} = \{0, 1, 2\}$. Then $\mathcal{A}^4$ consists of the 3^4 tuples of length 4 with entries $0, 1, 2$, like, for example,

$$0000,\ 0102,\ 2221 \text{ or } 2100.$$

A 3-ary code (also called **ternary**) of length 4 consists of a collection of such ternary 4-tuples. The Morse code from Section 1.1 really is a ternary code. The reason is that the individual letters need to be seperated. If we represent a dot by 0, a dash by 1 and a gap between letters as 2, the message SOS will be represented by the ternary word 00021112000.

1.9 Definition. *Let* $x = (x_1, x_2, \ldots, x_n)$ *and* $y = (y_1, y_2, \ldots, y_n)$ *be elements (strings, vectors, words) in* $\mathcal{A}^n$*. The* **distance** *(or* **Hamming distance***) between* x *and* y *is defined as*

$$d(x, y) = \textit{number of coordinates } i \textit{ where } x_i \neq y_i.$$

This is the same as in Definition 1.3. As before, the minimum distance of a code is the minimum of the distances between different codewords, and again the Hamming distance defines a metric.

1.10 Definition. *A q-ary code $(n, M, d)_q$ is a set of M strings in $\mathcal{A}^n$ (where $|\mathcal{A}| = q$) such that any two different elements of the code (codewords) are at distance $\geq d$. Call n the* **length** *and d the* **minimum distance** *of the code.*

Here are the words of a ternary code $(4, 27, 2)_3$:

0000	1002	2001	0102	1101	2100	0201	1200	2202
0012	1011	2010	0111	1110	2112	0210	1212	2211
0021	1020	2022	0120	1122	2121	0222	1221	2220

As in the binary case, in order to correct e errors, we need a code of minimum distance at least $2e + 1$. Our code $(4, 27, 2)_3$ will not suffice to correct one error. For example, if 0000 was sent and 0010 received (only one error occurred), the received vector has distance 1 not only from 0000 but also from 0012 and from 0210 and from 2010.

A basic problem of coding theory is the following: given q, n, d, find the maximum number M such that a code $(n, M, d)_q$ exists.

Error detection in telegraphy codes

Error detection is a more modest aim than error correction. It is suitable in situations where the channel is very good. On the rare occasions that an error occurs (and is detected), the receiver can then simply ask for retransmission. The alphabet of the telegraphy codes consists of the 26 letters A, B, ... , Z. The classical commercial codes use five letter groups as codewords. Each trade had its own elaborate codes. The primary aim of these codes was to save transmission time and thus to save money. As an example, take the **Acme Code.** It saves time to send the codeword BUKSI when *Avoid arrest if possible* is intended, and AROJD is shorter than *Please advertise the birth of twins.* It is a little unclear if PYTUO for *Collided with an iceberg* really achieves much in this respect, as such collisions do not happen all the time. In modern terminology, this business of representing messages by short strings is called **Data Compression** or **Source Coding.** It is not our concern in this book.

However, commercial telegraphy codes also took the reliability of message transmission into consideration. A general rule known as the **two-letter differential** stipulated that any two codewords had to differ in at least two letters. This means that each commercial code has minimum Hamming distance ≥ 2, enough to detect single errors. The Acme code also safeguarded

against a different type of error: no two codewords (necessarily of Hamming distance 2) may result from each other by transposition of two adjacent letters. For instance, if AHXNO is a codeword (it stands for *Met with a fatal accident* in the Acme code), then HAXNO, AXHNO, AHNXO, AHXON cannot be codewords.

This material is from Chapter 22 of D. Kahn's *The Codebreakers* [122].

1. A q-ary code $(n, M, d)_q$ is a collection of M q-ary n-tuples (the **codewords**) such that any two different codewords are at Hamming distance at least d.
2. A basic problem of coding theory: given q, n, d, maximize M such that an $(n, M, d)_q$-code exists.
3. A code can correct e errors provided its minimum distance is $d \geq 2e + 1$.
4. We saw a ternary code $(4, 27, 2)_3$.

Exercises 1.4

1.4.1. *Find the smallest length n such that an $(n, 27, 2)_3$ exists.*

1.4.2. *Prove the following: if there is an $(n, M, d)_q$, then there is an $(n + 1, M, d)_q$.*

1.4.3. *Prove the following: If there is an $(n, M, d)_{q-1}$, then there is an $(n, M, d)_q$.*

1.5 The binary symmetric channel

Basic concepts: The BSC, binomial numbers, subsets and paths, the Pascal triangle.

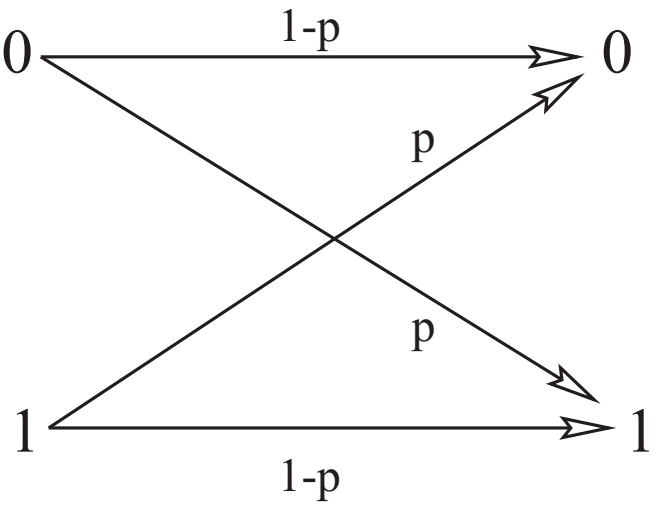

FIGURE 1.3: The BSC

So how do we model the **noise** mentioned in Section 1.1? In the case of the binary alphabet, the easiest and most widely used model is the binary symmetric channel (BSC). It assumes that there is a certain fixed probability, a number p, where $0 < p < 1$, for a bit transmission error. Clearly a value $p > 1/2$ does not make sense (why?) and $p = 1/2$ would mean that pure noise is received. We can therefore assume $p < 1/2$, and for all practical purposes p will be rather small.

The probability that a sent 0 will be received as 0 is $1 - p$. Likewise the probability that a sent 1 will be received as 1 is $1 - p$. The probability that a sent 0 is received as a 1 is p, just like the probability that a sent 1 is received as 0.

This model of a channel is called **symmetric** because 0 and 1 play symmetric roles. It is also called a **memoryless** channel because the probability of a bit error is independent of the prehistory. Can you think of situations when this model will not be appropriate?

In order to be able to do some combinatorial counting based on the BSC we take a look at the binomial numbers.

Binomials, subsets and paths

The number of bitstrings of length n and weight m is the binomial number $\binom{n}{m}$. This is the number of error patterns that can occur in n bits when the total number of errors is m. For example, the bitstrings of length 4 and weight 2 are

$$0011,\ 0110,\ 0101,\ 1001,\ 1010,\ 1100.$$

Accordingly, $\binom{4}{2} = 6$.

Another interpretation of the binomials uses subsets: identify each coordinate of a bitstring of length n with an element of a set, for example with the numbers $1, 2, \ldots, n$. Each bitstring of length n can then be identified with a subset of $\{1, 2, \ldots, n\}$. If the entry in the corresponding coordinate is 1, we include that element in the subset; if the entry is 0 we will not include it. For example, 0011 corresponds to the subset $\{3, 4\}$, 1001 to the subset $\{1, 4\}$, 1111 to the total set $\{1, 2, 3, 4\}$ and 0000 to the empty set. We see that the bitstrings of length n and weight m correspond precisely to the subsets of m elements of a fixed set with n elements.

A third interpretation of the binomials involves paths in a triangle; see Figure 1.4.

Consider paths starting at the top of the triangle, where in each step the choice is between going southeast or southwest. We may encode this decision by a string of E and W, for example EEWWE for going at first southeast twice, then southwest twice and a final step in southeast direction. Denote the top level by level 0. Then our path will end at level 5, at a node labelled 10 in Figure 1.4. Why that label? Our string with entries E and W is a bitstring in disguise. We can write 1 for E and 0 for W, obtaining bitstring 11001. Each path in the triangle is described by a bitstring. Bitstrings of length n end on level n. Two bitstrings end in the same spot if they have the same length and the same weight. This explains our labels: the label of the node on level n and weight m (start with weight 0 on the western end of the level, end with weight n on the eastern end) is the number of paths ending there. This is the number of bitstrings of length n and weight m, in other words the binomial $\binom{n}{m}$. The endpoint of our path is labeled 10, as $\binom{5}{3} = 10$ is the number of paths ending there. The labels 1 on the western border are the numbers $\binom{n}{0} = 1$ (there is only one bitstring with all entries 0 of any given length); the labels 1 on the eastern border are the numbers $\binom{n}{n} = 1$ (there is only one bitstring with all entries 1).

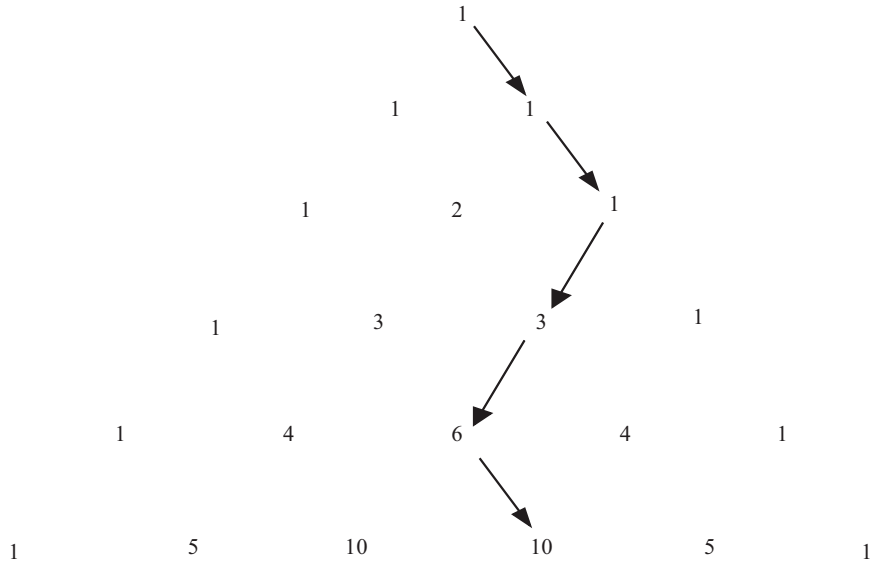

FIGURE 1.4: Pascal's triangle

The triangle is known as the **Pascal triangle.** It can be used to compute the binomials recursively. In fact, each label not on the border is the sum of the labels directly to its northeast and to its northwest, in formulas

$$\binom{n}{m} = \binom{n-1}{m-1} + \binom{n-1}{m} \text{ for } 0 < m < n.$$

A direct formula to compute the binomials is

$$\binom{n}{m} = \frac{n(n-1)\dots(n-m+1)}{m(m-1)\dots 2\cdot 1}.$$

As examples, $\binom{4}{2} = \frac{4\cdot 3}{2} = 6$, $\binom{5}{2} = \frac{5\cdot 4}{2} = 10$, $\binom{6}{3} = \frac{6\cdot 5\cdot 4}{3\cdot 2} = 20$ and in general $\binom{n}{2} = \frac{n(n-1)}{2}$.

Back to the channel

Assume a word (a bitstring) of length n is sent through the channel. The probability that no bit error occurs is $(1-p)^n$. The probability that exactly

one error occurs is $np(1-p)^{n-1}$ (there are n choices where the error could happen; the probability of a certain error pattern involving just one error is $p(1-p)^{n-1}$). The probability of a certain error pattern with precisely k errors is $p^k(1-p)^{n-k}$. The number of such error patterns is the number of possibilities of choosing k elements out of n. This is $\binom{n}{k}$. The probability that precisely k errors occur is therefore $\binom{n}{k}p^k(1-p)^{n-k}$. If we want to compute the probability that **at most** k errors happen, we have to sum up these probabilities.

1. The binary symmetric channel is the simplest model for noise.
2. If a binary n-tuple is sent via the BSC, the probability that at most k bit errors occur is $\sum_{i=0}^{k}\binom{n}{i}p^i(1-p)^{n-i}$.

While the BSC is conceptually simple, there are other channels which are easier to handle. An example is the binary **erasure channel** where the probability of correct transmission of 0 and 1 is $1-p$ just as in the BSC, but in addition the receiver knows when problems occurred. Formally this can be described as a channel with three possible outputs: $0, 1$ or E, where E stands for **erasure.**

Exercises 1.5

1.5.1. *Compute the probability that no more than one error occurs in the transmission of a bitstring of length* 10, *when the bit error probability is* $p = 10^{-3}$, *or* $p = 10^{-4}$, *or* $p = 10^{-5}$.

1.5.2. *Describe a generalization of the BSC from the binary to the general q-ary case.*

1.5.3. *Sketch a formal picture of the binary erasure channel, analogous to Figure 1.3 for the BSC.*

1.5.4. *Show that, if a code of minimum distance* d *is used for the erasure channel, then any* $d-1$ *errors can be corrected.*

1.5.5. *Use the subset interpretation of the binomials to prove the binomial formula*

$$(a+b)^n = \sum_{i=0}^{n} \binom{n}{i} a^i b^{n-i}.$$

1.5.6. *Prove*

$$(1-b)^n = \sum_{i=0}^{n} (-1)^i \binom{n}{i} b^i.$$

1.6 The sphere-packing bound

and the ternary Hamming code; the football pool problem.

As in all mathematical optimization problems, the problem of finding the maximum number M, such that a code $(n, M, d)_q$ exists, splits into two parts: we have to construct good codes (with high M), and we have to prove upper bounds showing that a higher value of M is impossible. In this section we will prove our first upper bound. It is a direct consequence of the basic idea of error correction.

As a preparation, we need a counting argument (this is a typical problem of elementary combinatorics) of the same type as the argument in Section 1.5. Given a vector $x \in \mathcal{A}^n$ (recall that $\mathcal{A}$ is our alphabet of size q), how many vectors are there at distance $\leq i$ from x? We call this set of vectors the **ball of radius** i with center x.

1.11 Definition. *Consider the space $\mathcal{A}^n$ of vectors of length n, with the Hamming metric, where $|\mathcal{A}| = q$. This is also called a* **Hamming space.** *The number of vectors at distance $\leq i$ from a given vector is denoted $V_q(i, n)$. We call $V_q(i, n)$ the* **volume** *of a ball of radius i.*

Fix some distance j and count the vectors at distance precisely j from the given vector. There are $\binom{n}{j}$ choices for the set of coordinates where the entries are different. Once this set is fixed, there are $q-1$ possibilities for the possible entries in each of these j coordinates. We count $\binom{n}{j}(q-1)^j$. The volume $V_q(i, n)$ is obtained by adding up these numbers, for $j \leq i$. We have seen the following:

1.12 Proposition. *The volume of the ball of radius i is*

$$V_q(i,n) = \sum_{j=0}^{i} \binom{n}{j} (q-1)^j.$$

We are close to our bound. Let $\mathcal{C}$ be a code $(n, M, d)_q$. Let $e = \lfloor (d-1)/2 \rfloor$ (the largest integer less than or equal to $(d-1)/2$). We have chosen e such that $2e + 1 < d$. Our standard argument shows that the balls of radius e centered at the codewords must be disjoint. As each such ball has $V_q(e,n)$ vectors, we must have $MV_q(e,n) \leq q^n$ (counting all vectors in these balls, we cannot get more vectors than the whole space contains).

1.13 Theorem (sphere packing bound)**.** *Each q-ary code $\mathcal{C}$ of length n and minimum distance d satisfies*

$$|\mathcal{C}| \leq \frac{q^n}{V_q(e,n)}.$$

Here $e = \lfloor (d-1)/2 \rfloor$.

Codes for which equality holds in Theorem 1.13 are known as **perfect codes.** The parameters of perfect codes have all been classified.
Our first code was a $(6, 8, 3)_2$. What is the maximum M for a code $(6, M, 3)_2$? Here $e = 1$. We have $V_2(1, 6) = 7$. The sphere-packing bound says $7M \leq 64$, hence $M \leq 9$. We claim that a code $(6, 9, 3)_2$ cannot exist. The reader is asked to provide a proof along the lines sketched in Exercises 1.6.4 and 1.6.5.

Here is a nice parameter situation: consider a possible code $(11, 729, 5)_3$. Observe that $729 = 3^6$. We have $e = 2$ and $V_3(2, 11) = 1 + 11 \cdot 2 + \binom{11}{2} \cdot 4 = 1 + 22 + 220 = 243 = 3^5$ (quite a coincidence). It follows that a ternary code $(11, M, 5)_3$ has $M \leq 3^{11}/3^5 = 3^6 = 729$. We see that the parameters $(11, 3^6, 5)_3$ are extremal. If such a code exists, then it is perfect. A perfect code $(11, 729, 5)_3$ does indeed exist. It is uniquely determined by its parameters and known as the **ternary Golay code,** named after the Swiss engineer who described it in 1949 (see Golay [94], reprinted in Berlekamp [13]).

A betting system

Here is how the ternary Golay code can be used in a betting system: in most European countries it is popular to bet on the results of football matches (**football** is not to be confused with American football. In the US the game is known by the bizarre name of **soccer**). Each match has one of three possible results: 1= home team wins, 2=guest wins, or 0= a draw. This is why we use ternary codes. There are 11 or more matches in the pool. If there are

11 matches, we can use the 729 codewords of the ternary Golay code as bets. As every vector is at distance ≤ 2 from a codeword, it is guaranteed that no matter what the results of the 11 matches are, one of our bets has at least 9 results correct. Assume there are 13 matches in the pool. Then we may choose two matches whose results seem to be safe and use the Golay code for the remaining 11 matches.

Incidentally, the ternary Golay code is older than Golay. The Finnish journalist Juhani Virtakallio published it in 1947 in the Finnish football journal *Veikaaja* (see Cohen et al. [55]). It is no coincidence that this discovery was made in Finland. This country has a rich tradition in coding theory and related questions.

1. The **sphere-packing bound** is our first general bound on codes.
2. Codes meeting it with equality are called **perfect.**
3. The **ternary Golay code** $(11, 729, 5)_3$ is perfect.

Exercises 1.6

1.6.1. *Construct a code* $(4, 2, 3)_2$ *(this is really trivial).*

1.6.2. *Show that there is no* $(4, 3, 3)_2$*-code*

1.6.3. *Construct a code* $(5, 4, 3)_2$ *(hint: use our code* $(6, 8, 3)_2$*)).*

1.6.4. *Show that there is no* $(5, 5, 3)_2$.

1.6.5. *Using the preceding exercise, show that there is no* $(6, 9, 3)_2$.

1.6.6. *What does the sphere-packing bound tell us about the length* n *of a binary code* $(n, 2^7, 5)_2$?

1.6.7. *Six candidates are examined by* 9 *referees. Each referee assigns a pass-fail grade to each candidate. Any two referees assign the same grade to not more than* 3 *of the candidates. Can this really happen?*

1.6.8. *Show that the minimum distance of a perfect code must be odd.*

1.6.9. *Use the sphere-packing bound to show the nonexistence of* $(5, 6, 3)_2$.

1.6.10. *Does a* $(7, 9, 3)_2$*-code exist?*

Taylor & Francis
Taylor & Francis Group
http://taylorandfrancis.com

Chapter 2

Binary linear codes

2.1 The concept of binary linear codes

Basic concepts: Dimension, generator matrices, minimum weight. Linear algebra over $\mathbb{F}_2$: basis, rank, linear independence, determinant.

Our binary code $(6, 8, 3)_2$ has an additional structure, which greatly simplifies its description. Consider the following three of its codewords:

100110
010101
001011

Call them z_1, z_2, z_3. Consider all **linear combinations** of z_1, z_2 and z_3, that is, all vectors of the form $\lambda_1 z_1 + \lambda_2 z_2 + \lambda_3 z_3$, where $\lambda_i \in \mathbb{F}_2$. These linear combinations are different and they are just exactly the words of our code. We call $\{z_1, z_2, z_3\}$ a **basis** of our code and say that the code is **linear.** Another way of seeing this is by the following observation: the sum of any two codewords is a codeword again; the code is closed under sums. We take this as a definition:

2.1 Definition. *A binary code is* **linear** *if it is closed under addition.*

It is a basic fact from linear algebra that each linear code (abstractly: each linear space, each vector space) has a basis. Linear algebra applies to arbitrary fields. Most people are familiar with fields like the rational numbers, the real numbers and the complex numbers, but we can apply the basics of linear algebra to finite fields like $\mathbb{F}_2$ as well. This leads to the following basic fact:

2.2 Theorem. *A binary linear code $(n, M, d)_2$ has $M = 2^k$ for some k. The number k is the* **dimension** *of the code. There is a* **basis** $\{z_1, z_2, \ldots, z_k\}$ *of k codewords. Each codeword is a linear combination of the z_i.*

There is a general tendency to restrict attention to **linear** codes. One reason is that these are much easier to describe and to work with than codes in general. For example, a binary linear code of dimension k has $M = 2^k$ codewords, but it is uniquely described by a basis, which has only k elements. This is a much more compact representation of the code than a list of all its words.

2.3 Definition. *The parameters of a k-dimensional binary linear code of length n and minimum distance d are written*

$$[n, k, d]_2$$

(the number of codewords is $M = 2^k$).

The basic problem of binary linear codes is the following: determine the maximum k such that a code $[n, k, d]_2$ exists. Our code $(6, 8, 3)_2$ is a linear code, a $[6, 3, 3]_2$-code. The compact representation described above leads to the notion of a generator matrix.

2.4 Definition. *Let $\mathcal{C}$ be a linear code $[n, k, d]_2$. A* **generator matrix** *G of $\mathcal{C}$ is a (k, n)-matrix whose rows form a basis of $\mathcal{C}$.*

If we know a generator matrix G, then we know the code. The codewords are just all linear combinations of the rows of G; in other words, the code is the rowspace of G.

Here is another binary linear code. Please check that it is indeed linear. The minimum distance is 3.

0000000	1100110
1101000	0100101
1010100	1000011
0110010	0001110
1110001	1001101
0111100	0101011
1011010	0010111
0011001	1111111

As there are $16 = 2^4$ codewords, its dimension is $k = 4$. The parameters are $[7, 4, 3]_2$. It is known as the **binary Hamming code.** How can we be sure the minimum distance is really 3? We would have to check $\binom{16}{2} = 120$ pairs of codewords. Here is a simplification, valid for linear codes:

2.5 Proposition. *For a binary linear code, the minimum distance equals the minimum of the weights of nonzero codewords (observe that the all-0 word is automatically contained in each linear code).*

PROOF We know that each weight also is a distance. It suffices to show that the distance between two codewords also is the weight of some codeword. Let x, y be different codewords at distance $d(x, y)$. Addition of the same vector to both words does not change the distance. We add x. This yields $d(x, y) = d(\mathbf{0}, y + x) = wt(y + x)$. Here $\mathbf{0}$ is the all-0 word. As the code is linear, $x + y$ is a codeword again. As $x \neq y$, we have $x + y \neq 0$.

Because of Proposition 2.5, it suffices to check that each of the 15 nonzero codewords has weight ≥ 3 to check that indeed $d = 3$. Here is a generator matrix:

$$\begin{pmatrix} 1\,0\,0\,1\,1\,0\,1 \\ 0\,1\,0\,1\,0\,1\,1 \\ 0\,0\,1\,0\,1\,1\,1 \\ 0\,0\,0\,1\,1\,1\,0 \end{pmatrix}$$

How can we be sure? At first we check that each row of the matrix is a nonzero codeword. One method of controlling that we have a generator matrix is to make sure that each codeword is indeed a linear combination of rows, equivalently that different linear combinations of rows yield different codewords. We can speed up by using basic facts of linear algebra. One such fact is the following: we have a generator matrix if and only if the matrix has rank equal to the number of rows ($= k$), if and only if there is a (k, k)-submatrix with nonzero determinant. In our case the last check is fastest: the first four columns form a triangular matrix with ones on the diagonal, hence of determinant $= 1$. We repeat:

2.6 Proposition. *Let $\mathcal{C}$ be a binary $[n, k]_2$-code (the minimum distance is irrelevant here). Let G be a matrix whose rows are codewords of $\mathcal{C}$. The following are equivalent:*

1. *G is a generator matrix.*

2. *G has rank k (remember that the* **rank** *of a matrix is a basic term from linear algebra).*

3. *There is a (k, k)-submatrix of nonzero determinant (recall that the* **determinant** *of a matrix is another basic notion from linear algebra).*

4. *The rows of G are* **linearly independent** *(yet another such basic notion).*

1. A binary code is **linear** (or a **subspace** of $\mathbb{F}_2^n$) if it is closed under addition.
2. A binary linear code (subspace) has 2^k codewords; k is the **dimension**.
3. The parameters of binary linear codes: $[n, k, d]_2$.
4. Each binary linear code has a **basis,** consisting of k codewords.
5. A (k, n)-matrix whose rows form a basis is a **generator matrix.**

Exercises 2.1

2.1.1. *Show that our code* $[7, 4, 3]_2$ *is perfect.*

2.1.2. *Try to decide if an* $[8, 4, 4]_2$ *exists.*

2.1.3. *Give an example showing that the basis of a code is not uniquely determined.*

2.1.4. *Determine the parameters of the binary linear code generated by the rows of the matrix*

$$\begin{pmatrix} 1\,0\,0\,1\,1\,0\,1 \\ 0\,1\,0\,1\,0\,1\,1 \\ 0\,0\,1\,0\,1\,1\,1 \end{pmatrix}$$

2.1.5. *Compute the parameters* $[n, k, d]_2$ *of the binary linear code generated by*

$$G = \begin{pmatrix} 1\,1\,0\,0\,0\,0\,1\,1\,0\,1 \\ 0\,0\,1\,1\,0\,0\,1\,0\,1\,1 \\ 0\,0\,0\,0\,1\,1\,0\,1\,1\,1 \end{pmatrix}$$

Find a nonzero codeword of minimum weight.

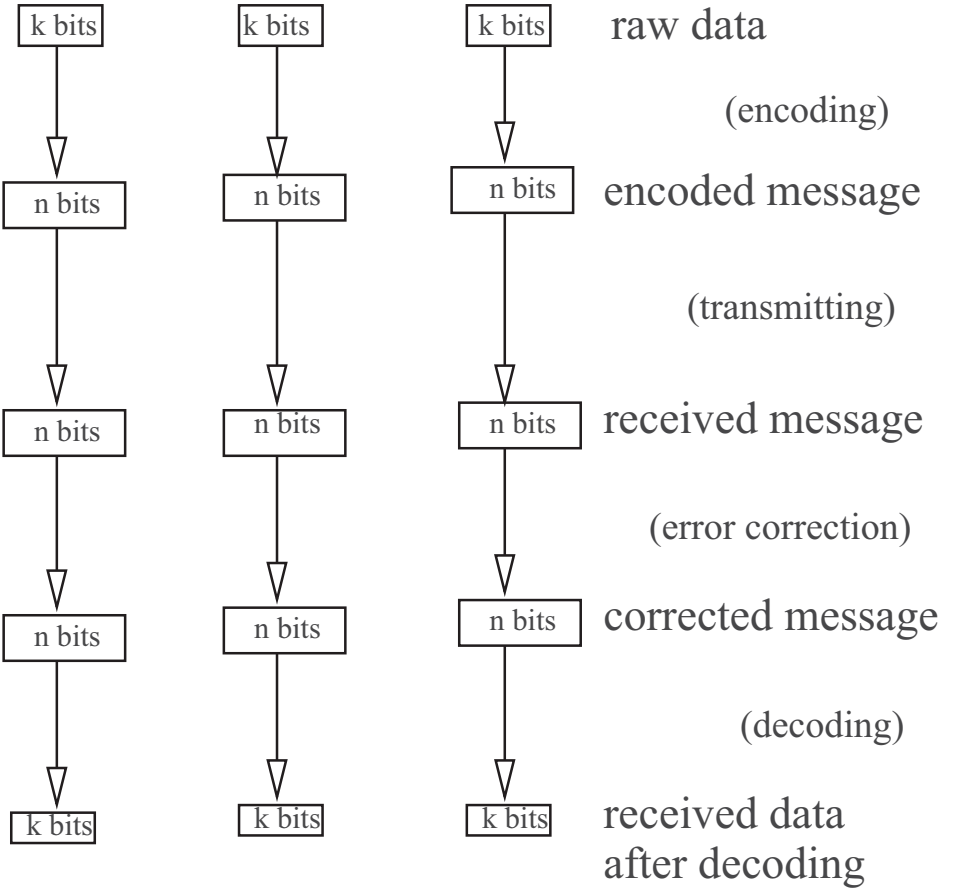

FIGURE 2.1: Block coding

2.2 Block coding

using binary linear codes. Information rate, relative distance.