

Information Theory, Coding and Cryptography

Third Edition

About the Author

Ranjan Bose is currently Professor in the Department of Electrical Engineering at the Indian Institute of Technology (IIT) Delhi. He obtained his BTech degree in Electrical Engineering from IIT Kanpur, Uttar Pradesh. He received MS and PhD degrees in Electrical Engineering from the University of Pennsylvania, Philadelphia, USA. He then worked at Alliance Semiconductor Inc., San Jose, California, USA as a Senior Design Engineer. Since November 1997, he has been associated with the Department of Electrical Engineering at IIT Delhi, where currently he is the Microsoft Chair Professor and also heads the Wireless Research Lab. His lectures on Wireless Communications form a part of the video courses offered by the National Program on Technology Enhanced Learning (NPTEL). Dr Bose also serves as the National Coordinator for the Mission Project on Virtual Labs, which enables students across the country to perform engineering lab experiments remotely. He is one of the founding members of Virtualwire Technologies, a start-up company incubated in IIT Delhi. Dr Bose has held the position of guest scientist at the Technical University of Darmstadt, Germany; University of Colorado, Boulder, USA; UNIK, Norway and University of Maryland, Baltimore County, USA. He has been delivering lectures frequently in the areas of information theory, coding and cryptography.

Dr Bose has been presented with the URSI Young Scientist award in 1999, Humboldt Fellowship in July 2000, Indian National Academy of Engineers (INAE) Young Engineers Award in 2003, AICTE Career Award for Young Teachers in 2004, the BOYSCAST Fellowship given by the Department of Science and Technology (DST) in 2005 and Dr Vikram Sarabhai Research Award for the year 2013. He also served as the Head of Bharti School of Telecom Technology and Management, IIT Delhi in the past and is the founding Head of the Center of Excellence in Cyber Systems and Information Assurance at IIT Delhi. He contributed significantly to *IETE Journal of Education* as the Editor-in-Chief and as an Editor of *Frequenz–Journal of RF-Engineering and Telecommunications*. Dr Bose is also a senior member of IEEE (USA) and a Fellow of IET (UK).

Information Theory, Coding and Cryptography

Third Edition

Ranjan Bose

*Professor, Department of Electrical Engineering
and*

*Head, Center of Excellence in Cyber Systems and Information Assurance
Indian Institute of Technology Delhi*



**McGraw Hill Education (India) Private Limited
NEW DELHI**

McGraw Hill Education Offices

New Delhi New York St Louis San Francisco Auckland Bogotá Caracas
Kuala Lumpur Lisbon London Madrid Mexico City Milan Montreal
San Juan Santiago Singapore Sydney Tokyo Toronto



McGraw Hill Education (India) Private Limited

Published by McGraw Hill Education (India) Private Limited
P-24, Green Park Extension, New Delhi 110 016

Information Theory, Coding and Cryptography, 3e

Copyright © 2016, 2008, 2003 by McGraw Hill Education (India) Private Limited.

No part of this publication may be reproduced or distributed in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise or stored in a database or retrieval system without the prior written permission of the publishers. The program listings (if any) may be entered, stored and executed in a computer system, but they may not be reproduced for publication.

This edition can be exported from India only by the publishers,
McGraw Hill Education (India) Private Limited.

Print Edition

ISBN (13 digit): 978-93-85880-56-8
ISBN (10 digit): 93-85880-56-X

Ebook Edition

ISBN (13 digit): 978-93-85880-57-5
ISBN (10 digit): 97-85880-57-8

Managing Director: *Kaushik Bellani*

Director—Products (Higher Education & Professional): *Vibha Mahajan*

Manager—Product Development: *Koyel Ghosh*

Specialist—Product Development: *Sachin Kumar*

Head—Production (Higher Education & Professional): *Satinder S Baveja*

Senior Copy Editor: *Kritika Lakhera*

Senior Manager—Production: *Piyaray Pandita*

Assistant General Manager—Product Management: *Shalini Jha*

Manager—Product Management: *Kartik Arora*

General Manager—Production: *Rajender P Ghansela*

Manager—Production: *Reji Kumar*

Information contained in this work has been obtained by McGraw Hill Education (India), from sources believed to be reliable. However, neither McGraw Hill Education (India) nor its authors guarantee the accuracy or completeness of any information published herein, and neither McGraw Hill Education (India) nor its authors shall be responsible for any errors, omissions, or damages arising out of use of this information. This work is published with the understanding that McGraw Hill Education (India) and its authors are supplying information but are not attempting to render engineering or other professional services. If such services are required, the assistance of an appropriate professional should be sought.

Typeset at Tej Composers, WZ 391, Madipur, New Delhi 110 063 and printed at

Visit us at: www.mheducation.co.in

*To Dear Ma,
My First Teacher*

Contents

<i>Preface to the Third Edition</i>	xv
<i>Preface to the First Edition</i>	xxiii
<i>List of Abbreviations</i>	xxv
0. Pre-requisite	1

0.1 Introduction to Matrices	1
0.2 Introduction to Probability Theory	4

Part I Information Theory and Source Coding

1. Source Coding	11
-------------------------	-----------

<i>Learning Objectives</i>	11
----------------------------	----

1.1 Introduction to Information Theory	11
1.2 Uncertainty and Information	13
1.3 Average Mutual Information and Entropy	18
1.4 Information Measures for Continuous Random Variables	22
1.5 Relative Entropy	23

<i>Learning Review</i>	25
------------------------	----

1.6 Source Coding Theorem	26
---------------------------	----

<i>Learning Review</i>	31
------------------------	----

1.7 Huffman Coding	31
--------------------	----

1.8 Shannon-Fano-Elias Coding	39
-------------------------------	----

1.9 Arithmetic Coding	40
-----------------------	----

1.10 The Lempel-Ziv Algorithm	42
-------------------------------	----

1.11 Run Length Encoding	44
--------------------------	----

<i>Learning Review</i>	46
------------------------	----

1.12 Rate Distortion Function	47
-------------------------------	----

1.13 Optimum Quantizer Design	49
-------------------------------	----

1.14 Entropy Rate of a Stochastic Process	50
---	----

<i>Learning Review</i>	53
------------------------	----

1.15	Introduction to Image Compression	54
1.16	The JPEG Standard for Lossless Compression	55
1.17	The JPEG Standard for Lossy Compression	56
1.18	Video Compression Standards	57
	<i>Learning Review</i>	60
1.19	Concluding Remarks	60
	<i>Learning Outcomes</i>	61
	<i>Multiple Choice Questions</i>	64
	<i>Short-Answer Type Questions</i>	65
	<i>Problems</i>	66
	<i>Computer Problems</i>	69
	<i>Project Ideas</i>	70
	<i>References for Further Reading</i>	71
2.	Channel Capacity and Coding	72
	<i>Learning Objectives</i>	72
2.1	Introduction	72
2.2	Channel Models	74
	<i>Learning Review</i>	77
2.3	Channel Capacity	78
	<i>Learning Review</i>	82
2.4	Channel Coding	82
	<i>Learning Review</i>	87
2.5	Information Capacity Theorem	87
2.6	Parallel Gaussian Channels	93
2.7	The Shannon Limit	95
2.8	Channel Capacity for MIMO Systems	97
2.9	Capacity Region for Multiple Access Channels	98
	<i>Learning Review</i>	100
2.10	Random Selection of Codes	100
	<i>Learning Review</i>	106
2.11	Concluding Remarks	107
	<i>Learning Outcomes</i>	107
	<i>Multiple Choice Questions</i>	109
	<i>Short-Answer Type Questions</i>	110
	<i>Problems</i>	110
	<i>Computer Problems</i>	114
	<i>Project Ideas</i>	114
	<i>References for Further Reading</i>	115

Part II Error Control Coding (Channel Coding)

3. Linear Block Codes for Error Correction 119

Learning Objectives 119

3.1 Introduction to Error Correcting Codes 119

3.2 Basic Definitions 121

3.3 Matrix Description of Linear Block 127

3.4 Equivalent Codes 128

3.5 Parity Check Matrix 130

Learning Review 133

3.6 Decoding of a Linear Block Code 134

3.7 Syndrome Decoding 140

3.8 Error Probability after Coding (Probability of Error Correction) 141

Learning Review 144

3.9 Perfect Codes 145

3.10 Hamming Codes 147

3.11 Low Density Parity Check (LDPC) Codes 149

3.12 Optimal Linear Codes 152

3.13 Maximum Distance Separable (MDS) Codes 153

3.14 Bounds on Minimum Distance 153

Learning Review 155

3.15 Space Time Block Codes 155

Learning Review 159

3.16 Concluding Remarks 159

Learning Outcomes 160

Multiple Choice Questions 162

Short-Answer Type Questions 163

Problems 164

Computer Problems 167

Project Ideas 167

References for Further Reading 168

4. Cyclic Codes 169

Learning Objectives 169

4.1 Introduction to Cyclic Codes 169

4.2 Polynomials 170

4.3 The Division Algorithm for Polynomials 172

Learning Review 176

4.4 A Method for Generating Cyclic Codes 177

4.5 Matrix Description of Cyclic Codes 180

4.6 Quasi-Cyclic Codes and Shortened Cyclic Codes 184

4.7 Burst Error Correction 185

<i>Learning Review</i>	186
4.8 Fire Codes	187
4.9 Golay Codes	188
4.10 Cyclic Redundancy Check (CRC) Codes	189
<i>Learning Review</i>	192
4.11 Circuit Implementation of Cyclic Codes	193
<i>Learning Review</i>	198
4.12 Concluding Remarks	198
<i>Learning Outcomes</i>	199
<i>Multiple Choice Questions</i>	201
<i>Short-Answer Type Questions</i>	202
<i>Problems</i>	203
<i>Computer Problems</i>	204
<i>Project Ideas</i>	205
<i>References for Further Reading</i>	206
5. Bose–Chaudhuri Hocquenghem (BCH) Codes	207
<i>Learning Objectives</i>	207
5.1 Introduction to BCH Codes	207
5.2 Primitive Elements	208
5.3 Minimal Polynomials	211
<i>Learning Review</i>	215
5.4 Generator Polynomials in Terms of Minimal Polynomials	215
5.5 Some Examples of BCH Codes	216
5.6 Decoding of BCH Codes	220
<i>Learning Review</i>	225
5.7 Reed-Solomon Codes	225
5.8 Implementation of Reed-Solomon Encoders and Decoders	228
5.9 Performance of RS Codes Over Real Channels	230
5.10 Nested Codes	233
<i>Learning Review</i>	235
5.11 Concluding Remarks	235
<i>Learning Outcomes</i>	236
<i>Multiple Choice Questions</i>	237
<i>Short-Answer Type Questions</i>	238
<i>Problems</i>	239
<i>Computer Problems</i>	241
<i>Project Ideas</i>	241
<i>References for Further Reading</i>	242
6. Space–Time Codes	243
<i>Learning Objectives</i>	243
6.1 Introduction to Space-Time Codes	243

6.2	Anatomy of Space-Time Block Code	244
6.3	Space-Time Code Design Criteria	248
<i>Learning Review</i>		251
6.4	Real Orthogonal Design	252
6.5	Generalised Real Orthogonal Design	253
6.6	Complex Orthogonal Design	256
6.7	Quasi-Orthogonal Space-Time Block Code	260
<i>Learning Review</i>		262
6.8	STBC Design Targets and Performance	263
<i>Learning Review</i>		266
6.9	Concluding Remarks	267
<i>Learning Outcomes</i>		267
<i>Multiple Choice Questions</i>		268
<i>Short-Answer Type Questions</i>		270
<i>Problems</i>		271
<i>Computer Problems</i>		273
<i>Project Ideas</i>		273
<i>References for Further Reading</i>		274

Part III Codes on Graph

7.	Convolutional Codes	277
<i>Learning Objectives</i> 277		
7.1	Introduction to Convolutional Codes	277
7.2	Tree Codes and Trellis Codes	278
7.3	Polynomial Description of Convolutional Codes (Analytical Representation)	283
<i>Learning Review</i> 288		
7.4	Distance Notions for Convolutional Codes	289
7.5	The Generating Function	291
7.6	Matrix Description of Convolutional Codes	293
7.7	Viterbi Decoding of Convolutional Codes	295
7.8	Distance Bounds for Convolutional Codes	301
7.9	Performance Bounds	303
7.10	Known Good Convolutional Codes	304
<i>Learning Review</i> 307		
7.11	Turbo Codes	308
7.12	Turbo Decoding	310
7.13	Interleaver Design for Turbo Codes	315
<i>Learning Review</i> 316		
7.14	Concluding Remarks	316
<i>Learning Outcomes</i> 317		
<i>Multiple Choice Questions</i> 319		
<i>Short-Answer Type Questions</i> 320		

<i>Problems</i>	321
<i>Computer Problems</i>	324
<i>Project Ideas</i>	326
<i>References for Further Reading</i>	326

8. Trellis Coded Modulation	327
------------------------------------	------------

<i>Learning Objectives</i>	327
8.1 Introduction to TCM	327
8.2 The Concept of Coded Modulation	328
8.3 Mapping by Set Partitioning	333
8.4 Ungerboeck's TCM Design Rules	336
<i>Learning Review</i>	341
8.5 TCM Decoder	342
8.6 Performance Evaluation for AWGN Channel	343
8.7 Computation of d_{free}	348
8.8 TCM for Fading Channels	349
<i>Learning Review</i>	353
8.9 Space Time Trellis Codes	353
<i>Learning Review</i>	357
8.10 Concluding Remarks	357
<i>Learning Outcomes</i>	358
<i>Multiple Choice Questions</i>	360
<i>Short-Answer Type Questions</i>	361
<i>Problems</i>	362
<i>Computer Problems</i>	366
<i>Project Ideas</i>	367
<i>References for Further Reading</i>	368

Part IV Coding for Secure Communications

9. Cryptography	371
------------------------	------------

<i>Learning Objectives</i>	371
9.1 Introduction to Cryptography	371
9.2 An Overview of Encryption Techniques	373
9.3 Operations used by Encryption Algorithms	375
<i>Learning Review</i>	378
9.4 Symmetric (Secret Key) Cryptography	379
9.5 Data Encryption Standard (DES)	382
9.6 International Data Encryption Algorithm (IDEA)	385
9.7 RC Ciphers	386
<i>Learning Review</i>	387
9.8 Asymmetric (Public-Key) Algorithms	387
9.9 The RSA Algorithm	388

9.10	Pretty Good Privacy (PGP)	393
9.11	One-way Hashing	395
9.12	Other Techniques	397
	<i>Learning Review</i>	398
9.13	Elliptic Curve Cryptography	398
9.14	Diffie-Hellman Key Agreement Protocol	400
9.15	Secure Communication using Chaos Functions	402
9.16	Quantum Cryptography	403
9.17	Biometric Encryption	404
9.18	Cryptanalysis	405
9.19	Politics of Cryptography	407
	<i>Learning Review</i>	408
9.20	Concluding Remarks	409
	<i>Learning Outcomes</i>	412
	<i>Multiple Choice Questions</i>	414
	<i>Short-Answer Type Questions</i>	415
	<i>Problems</i>	416
	<i>Computer Problems</i>	419
	<i>Project Ideas</i>	419
	<i>References for Further Reading</i>	420

10.	Physical Layer Security	421
	<i>Learning Objectives</i>	421
10.1	Introduction to Physical Layer Security	421
10.2	Shannon's Notion of Security	422
10.3	The Wiretap Model	423
	<i>Learning Review</i>	431
10.4	The Gaussian Wiretap Model	432
10.5	Secrecy Capacity in Wireless Channels	434
	<i>Learning Review</i>	438
10.6	Cooperative Jamming	439
10.7	Artificial Noise Forwarding	440
	<i>Learning Review</i>	442
10.8	Concluding Remarks	443
	<i>Learning Outcomes</i>	443
	<i>Multiple Choice Questions</i>	445
	<i>Short-Answer Type Questions</i>	447
	<i>Problems</i>	447
	<i>Computer Problems</i>	451
	<i>Project Ideas</i>	452
	<i>References for Further Reading</i>	453

Preface to the Third Edition

Introduction to the Course

The textbook is designed for the students of electrical/electronics engineering and computer science. The primary aim of this book is to arouse the curiosity of the students. This edition is prepared, keeping in mind, the current needs of students and instructors. The revised text is (hopefully) more fascinating-to-read, simple-to-understand, logical-to-follow and motivating-to-design. An attempt has been made to adhere to the original philosophy behind writing this book—it is to be a lively introduction to the topics dealing with *Information Theory, Coding and Cryptography*.

Target Audience

This book is intended for final-year undergraduate students and first-year postgraduate students of the electrical engineering or the computer science programs. The book will help in forming a strong foundation for the broad areas of *Information Theory, Coding and Cryptography*. It emphasizes on the basic concepts, lays stress on the fundamental principles and motivates their application to practical problems. By design, the mathematical complexity of the book remains at a level well within the grasp of engineering college students. The book can also be used as a quick reference by practicing engineers.

Objectives of the Revision

The main objectives of the revision are to update the material of the previous edition, to make it easier to teach and to add **Learning Objectives** in each chapter. Two new chapters have also been added: **Chapter 6-Space Time Codes** and **Chapter 10-Physical Layer Security**. In addition to these, several other interesting topics have been included, for example, Relative Entropy, Video Compression Standards, Parallel Gaussian Channels and Capacity region for Multiple Access Channels. While making these modifications, the main aim was to increase the utility-quotient of this book for the students and instructors.

Roadmap of Target Courses

This book is ideally suited for a full semester-long course at the post-graduate level. For an advanced undergraduate course, the following topics may be omitted: Relative Entropy, Rate Distortion Function, Channel capacity for MIMO systems, Capacity region for Multiple Access Channels, Bounds on Minimum Distance, Circuit Implementation of Cyclic Codes, Performance of RS codes over real channels, Nested Codes, Quasi-Orthogonal Space Time Block Codes, Turbo decoding, Interleaver Design for Turbo Codes, TCM for Fading Channels, Cooperative Jamming and Artificial Noise Forwarding. Certain portions of this book can also be used for short-term courses, quality-improvement programs and continuing education programs.

What's New in the Third Edition?

The third edition is student-centric, and follows the ‘Learning Objective/Levels of Difficulty (LO/LOD)’ approach. This is an educational process that emphasizes on developing engineering skills in student and testing the outcomes of the study of a course, as opposed to rote learning. Each of the 10 chapters follows a common organizational structure with a range of learning and assessment tools for instructors and students. The feedback received on the second edition has been constructively used to augment the book in different dimensions. Following are the specific improvements over the earlier editions:

Learning Tools

➤ Use of Technology

We have taken advantage of recent technological developments to create a wealth of useful information not present in the physical book. For students using smartphones and tablets, scanning **QR codes** located within the chapters gives them immediate access to resources such as

- Introductory Video
- Answers to Learning Review and Short-Answer Type Questions
- Interactive Quizzes



*The answers are available here. Scan to check
Or
Visit <http://qrcode.flipick.com/index.php/586>*



CHAPTER ONE

Source Coding

Not everything that can be counted counts, and not everything that counts can be counted.
Albert Einstein (1879-1955)

learning objectives

After studying this chapter the students will be able to:

- LO 1** Understanding the concept of information.
- LO 2** State and prove the source coding theorem and discuss its ramifications.
- LO 3** Analyze five source coding techniques: the Huffman encoding, Shannon-Fano-Elias encoding, Arithmetic encoding, the Lempel-Ziv encoding and Run Length encoding.
- LO 4** Underline the concept of the rate distortion function and the design of the optimum quantizer.
- LO 5** Apply the knowledge to study image compression, one of the important application areas of source coding.

➤ Learning Objectives

Every chapter starts with a clear-cut list of learning objectives which are closely linked to the topics covered in the chapter. Each section within the chapter is tied to a specific learning objective. This will help the students better anticipate what they will be learning and the instructors to measure student’s understanding.

➤ Arrangement of Pedagogy

The Pedagogy is arranged as per levels of difficulty—All questions and problems are linked with Learning Objectives (LOs) and marked with Levels of Difficulty (LOD), to help students better assess their learning. This assessment of levels of difficulty is derived from the Blooms' Taxonomy.

- S** indicate Level 1 and Level 2 i.e., Knowledge and Comprehension based easy-to-solve problems
- M** indicate Level 3 and Level 4 i.e., Application and Analysis based medium-difficulty problems
- D** indicate Level 5 and Level 6 i.e., Synthesis and Evaluation based high-difficulty problems.

Levels of Difficulty

- S Simple:** Level 1 and Level 2 Category
- M Medium:** Level 3 and Level 4 Category
- D Difficult:** Level 5 and Level 6 Category

The units of $I(x)$ are determined by the base of the logarithm, which is usually selected as 2 or e . When the base is 2, the units are in **bits** and when the base is e , the units are in **nats** (natural units). Since $0 \leq P(x) \leq 1$, $I(x) \geq 0$, i.e., self-information is *non-negative*. The following two examples illustrate why a logarithmic measure of information is appropriate.

DID YOU KNOW Information has become a key to success (it has always been the key to success, but in today's world it is *the* key). And behind all this exchange of information lie the tiny 1's and 0's (the omnipresent bits) that hold the information by the mere way they sit next to one another. Yet the present day's information age owes primarily to a seminal paper published in 1948 that laid the foundation of the wonderful field of **Information Theory**—a theory that was initiated by one man, the American Electrical Engineer Claude E. Shannon, whose ideas appeared in the article "The Mathematical Theory of Communication" in the *Bell System Technical Journal* (1948). In its broadest sense, information is interpreted to include the message, television, and data-processing humans and o

INDUSTRIAL RELEVANCE Low-bit-rate speech coding is used both for communication and voice storage applications. Typical rates are below 4 kbit/s. At such low rates, full encoding of the speech waveform is not possible. Therefore, it becomes relevant to use parametric models to represent only those perceptually relevant aspects of speech. At bit rates above 4 kbit/s, speech-specific waveform coders based on code excited linear prediction (CELP) is used to produce good quality speech. Low-bit-rate speech coding is used in U.S. MIL-STD 3005 2.4 kbit/s MELP and the NATO STANAG 1591 MELP coding suite at 2400, 1200, and 600 b/s and the ITU 4 kb/s hybrid MELP/CELP speech coder.

➤ Highlighting Useful Information

To keep the readers engrossed, keywords such as **Don't Forget**, **Take a Look**, **Did You Know**, **Intuition**, **Interpretation** and **Industrial Relevance** have been interspersed throughout the text to draw attention to important anecdotes.

➤ Learning Outcomes

Every chapter ends with a telegraphic summary, where all the important concepts and formulae are compiled. This will enable the students to quickly and efficiently revise the key concepts.

LEARNING OUTCOMES

- ☒ A subfield of a field is a subset of the field, which is also a field. Thus, the subfield is 'contained' in the original field. Alternately, we can say that there is a base field, which is contained in the extension field. The base field is also called a ground field.
- ☒ A primitive element of $GF(q)$ is an element α such that every field element, except zero, can be expressed as a power of α . A field can have more than one primitive element.
- ☒ For an element $\beta \in GF(q)$, let n be the smallest integer such that $\beta^n = 1$, then n is the order of β . The order of a primitive element is $q - 1$.
- ☒ A primitive polynomial $p(x)$ over $GF(q)$ is a prime polynomial over $GF(q)$ with the property that in the extension field constructed modulo $p(x)$, the field element represented by x is a primitive element.
- ☒ A blocklength n of the form $n = q^m - 1$ is called a primitive block length for a code over $GF(q)$. A cyclic code over $GF(q)$ of primitive blocklength is called a primitive cyclic code.
- ☒ It is possible to factor $x^{q^m-1} - 1$ in the extension field $GF(q^m)$ to get $x^{q^m-1} - 1 = \prod_j (x - \beta_j)$, where β_j ranges over all the non-zero elements of $GF(q^m)$. This implies that each of the polynomials $f_i(x)$ can be represented in $GF(q^m)$ as a product of some of the linear terms, and each β_j is a zero of exactly one of the $f_i(x)$. This $f_i(x)$ is called the minimal polynomial of β_j .
- ☒ Suppose q is a prime, then an irreducible m^{th} degree polynomial, $f(x)$ with coefficients in $GF(q)$ divides $x^{q^m-1} - 1$.
- ☒ Two elements of $GF(q^m)$ that share the same minimal polynomial over $GF(q)$ are called conjugates with respect to $GF(q)$.

Integrated Assessment Tools

➤ In-text Examples

An example is a very effective teaching tool for illustrating a particular concept. Whenever a new concept is introduced in a chapter, it is followed up with an example. The revised edition now contains over 210 solved examples to facilitate the absorption of various concepts.

Example 1.1 Consider a binary source which tosses a fair coin and produces an output equal to 1 if a head appears and a 0 if a tail appears. For this source, $P(1) = P(0) = 0.5$. The information content of each output from the source is

$$I(x_i) = -\log_2 P(x_i) = -\log_2(0.5) = 1 \text{ bit} \quad (1.2)$$

Indeed, we have to use only one bit to represent the output from this binary source (say, we use a 1 to represent H and a 0 to represent T). Now, suppose the successive outputs from this binary source are statistically independent, *i.e.*, the source is memoryless. Consider a block of m binary digits. There are 2^m possible m -bit blocks, each of which is equally probable with probability 2^{-m} . The self information of an m -bit block is

$$I(x_i) = -\log_2 P(x_i) = -\log_2 2^{-m} = m \text{ bits} \quad (1.3)$$

Again, we observe that we indeed need m bits to represent the possible m -bit blocks. Thus, this logarithmic measure of information possesses the desired additive property when a number of source outputs are considered as a block.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 3:

1. Consider the Turbo encoder for in 3GPP-LTE given in Fig. 7.40. Let the interleaver be ‘flip all bits’. Encode the bit stream: 0 0 1 1 0 1 1 0 ... using the Turbo encoder.
2. Again, consider the Turbo encoder for in 3GPP-LTE given in Fig. 7.40. Let the interleaver be ‘first in last out’ of size 8 bits. Encode the bit stream: 1 0 0 1 0 0 0 ... using the Turbo encoder.

If you have successfully solved the above problems,
you have mastered LO 3. Congratulations!



➤ In-text Exercises

Linked to each Learning Objective (LO), there is a set of in-text exercises, marked with Levels of Difficulty (LOD). This assessment of levels of difficulty is derived from Bloom’s Taxonomy.

➤ Multiple Choice Questions

At the end of each chapter, there is a set of Multiple Choice Questions to test higher order thinking skills (HOTS) of the students.

➤ Short-Answer Type Questions

At the end of each chapter, there is also a set of Short-Answer Type Questions that can be used by students to discover ‘what they know’ and ‘what they don’t know’.



MULTIPLE CHOICE QUESTIONS

- 1.1 Pick the correct choice
 - (a) $I(X; Y) = H(X) - H(X|Y)$
 - (b) $I(X; Y) = H(Y) - H(Y|X)$
 - (c) Both a and b
 - (d) None
- 1.2 Which among the following is used to construct the binary code that satisfies the prefix condition?
 - (a) Information Rate
 - (b) Noisless Channel
 - (c) Channel Coding Theorem
 - (d) Kraft Inequality
- 1.3 The efficiency of a prefix code is given by
 - (a) $\eta = \frac{\bar{R}}{H(X)}$
 - (b) $\eta = \frac{H(X)}{\bar{R}}$
 - (c) $\eta = \frac{H(X) + 1}{\bar{R}}$
 - (d) None of the above
- 1.4 The Kullback Leibler (KL) distance violates which property of a distance measure
 - (a) Non-negative
 - (b) Symmetry property
 - (c) Triangle inequality
 - (d) None of the above
- 1.5 The relationship between mutual information and joint entropy can be expressed as
 - (a) $I(X; Y) = H(X) - H(X|Y)$
 - (b) $I(X; Y) = H(Y) - H(Y|X)$
 - (c) $I(X; Y) = H(X) + H(Y) - H(X, Y)$
 - (d) None of the above
- 1.6 The expected codeword length of Huffman code is bounded as
 - (a) $\bar{R} \leq H(X) + 1$
 - (b) $\bar{R} \leq H(X) - 1$
 - (c) $\bar{R} < H(X) + 1$
 - (d) None of the above

SHORT-ANSWER TYPE QUESTIONS

- 1.1 Can differential entropy be negative? Explain.
- 1.2 We require infinite number of bits to represent a continuous random variable precisely. True or false?
- 1.3 If X and Y are independent, is it true that $H(X, Y) = H(X) + H(Y)$?
- 1.4 Let X , Y , and Z be discrete random variables with $I(X; Y) = 0$ and $I(X; Z) = 0$. Is $I(X; Y, Z) = 0$?



**PROBLEMS**

- M** 1.1 Prove that the entropy for a discrete source is a maximum when the output symbols are equally probable.
- S** 1.2 Prove the inequality $\ln x \leq x - 1$. Plot the curves $y_1 = \ln x$ and $y_2 = x - 1$ to demonstrate the validity of this inequality.
- M** 1.3 Show that $I(X; Y) \geq 0$. Under what condition does the equality hold?
- S** 1.4 Let X denotes the number of tosses required for a coin until the first tail appears.
- (i) Find the entropy, $H_f(X)$ if the coin is fair.
 - (ii) Next assume the coin to be unfair with p being the probability of getting a tail. Find the entropy, $H_g(X)$.
- S** 1.5 Consider a code that satisfies the suffix condition: No codeword is a suffix of any other codeword.
- (i) Is this a uniquely decodable code? Why?
 - (ii) Can you relate the minimum average length of suffix codes to the average length of the Huffman code for the same random variable? Explain.

> Enhanced set of Chapter-end Problems

The revised edition now contains close to 200 chapter-end problems. These would strengthen students' understanding of the material and offer new perspective.

**COMPUTER PROBLEMS**

- 1.1 Write a program that performs Huffman coding, given the source probabilities. It should generate the code and give the coding efficiency. It should be able to handle non-binary inputs as well.
- 1.2 Modify the above program so that it can group together n source symbols and then generate the Huffman code. Plot the coding efficiency η versus n for the following source symbol probabilities: $\{0.55, 0.25, 0.20\}$. For what value of n does the efficiency become better than 0.9999? Repeat the exercise for the following source symbol probabilities $\{0.45, 0.25, 0.15, 0.10, 0.05\}$.
- 1.3 Write a program that executes the Lempel-Ziv algorithm. The input to the program can be the English alphabets. It should convert the alphabets to their ASCII code and then perform the compression routine. It should output the compression achieved. Using this program, find out the compression achieved for the following strings of letters:
- (i) The Lempel-Ziv algorithm can compress the English text by about fifty five percent.
 - (ii) The cat can not sit on the canopy of the car.

PROJECT IDEAS

- 1.1 Superinformation:** We wish to determine the 'entropy of entropy', which we define as *superinformation*.
- (i) Generate a 'long string' of characters randomly. The number of characters and their probabilities of occurrences are design parameters.
 - (ii) Divide the string into sub-blocks of size B .
 - (iii) For i^{th} sub-block, find the entropy, H_i .
 - (iv) Find the histogram of H_i . Normalize this histogram to obtain a pdf.
 - (v) Find the entropy for this pdf. This is the superinformation (entropy of entropy) of the sequence.
 - (vi) Use the concept of superinformation on DNA sequences and see if you can segregate exons (coding regions) from introns (non-coding regions).
- 1.2 Efficiency of the Morse Code:** The Morse Code for the English alphabet is given in Fig. 1.21.
- (i) Determine the average codeword length for the Morse Code and determine the efficiency of the code.
 - (ii) Is it uniquely decodable? Explain.

> Project Ideas

Another interesting modification is the inclusion of project ideas in each chapter, which can be taken up by students, either individually or under faculty supervision. The revised edition contains over 40 challenging project ideas.

> References for Further Reading

For the enthusiastic students, references for further reading have been provided at the end of each chapter.

REFERENCES FOR FURTHER READING

1. Roth, R.M., *Introduction to Coding Theory*, Cambridge University Press, 2006.
2. Lin, S. and Costello, D.J., *Error Control Coding: Fundamentals and Applications* (2nd edition), Prentice-Hall, 2004.
3. Blahut, R.E., *Algebraic Codes for Data Transmission*, Cambridge University Press, 2002.
4. Moon, T.K., *Error Correction Coding: Mathematical Methods and Algorithms*, Wiley, 2005.
5. Morelos-Zaragoza, R.H., *The Art of Error Correcting Coding*, Wiley & Sons, 2006.

Organisation of the Book

This edition of the book retains the original flavor. It has been divided into four logical parts which are as follows:

Chapter 0 in the book covers **pre-requisites**. It gives an introduction to Matrices and Probability Theory by providing important definitions relevant to the topics that have been introduced herein this chapter.

Part I: Information Theory and Source Coding

The first part consists of following two fundamental chapters:

Chapter 1 deals with information and its efficient representation. It introduces the important concepts of Source Coding, Huffman Coding, Shannon-Fano-Elias Coding, Arithmetic Coding, Lempel-Ziv Coding, Run Length Coding and the Rate Distortion Theory. Image and Video Compression techniques are also covered in this chapter.

Chapter 2 deals with Channel Capacity and Coding. The reader is introduced to various types of channel models. Information Capacity Theorem, Gaussian Channels, Channel capacity for MIMO systems, Capacity region for Multiple Access Channels and Random selection of codes are also discussed in detail. This chapter also motivates the need for Error Control Coding, which is covered in Part II.

Part II: Error Control Coding (Channel Coding)

This part consists of the following four interesting chapters:

Chapter 3 introduces the reader to the fascinating world of Linear Block Codes for Error Correction. Encoding and decoding of linear block codes are discussed in detail. Ideas related to Syndrome decoding, Perfect codes, Optimal linear codes and Maximum distance separable (MDS) Codes are presented in the chapter. Hamming Codes and Low Density Parity Check (LDPC) Codes are also discussed.

Chapter 4 deals with Cyclic Codes, a useful sub-class of linear block codes. This chapter introduces the Matrix description of cyclic codes, Quasi-cyclic codes, Shortened cyclic codes, Fire Codes, Golay Codes and the powerful Cyclic Redundancy Check (CRC) Codes. Circuit implementation of Cyclic Codes has also been discussed here.

Chapter 5 deals with Bose–Chaudhuri Hocquenghem (BCH) Codes, a useful sub-class of cyclic codes. The chapter builds the necessary mathematical background required to work with BCH codes. Reed Solomon (RS) Codes, an important sub-class of BCH codes, are also discussed, along with their performance over real channels.

Chapter 6 is a new chapter and introduces the reader to the intriguing world of Space-Time Codes. The chapter covers Real Orthogonal Design, Generalized Real Orthogonal Design, Complex Orthogonal Design, Generalized Complex Orthogonal Design and Quasi-orthogonal Space Time Block Codes. Space-Time Code design criteria and design targets are also discussed in this chapter.

Part III: Codes on Graph

This part consists of following two fascinating chapters that deal with codes with memory:

Chapter 7 discusses various aspects of Convolutional Codes, including the Polynomial description, Distance Notion, Generating Function and the Matrix description. Viterbi decoding of Convolutional Codes is discussed in detail. Turbo Codes, Turbo decoding and Interleaver Design for Turbo Codes are also covered in this chapter.

Chapter 8 deals with Combined Coding and Modulation. The basic idea behind Trellis Coded Modulation (TCM) is introduced. Mapping by set partitioning, Ungerboeck's TCM Design Rules, TCM for AWGN Channels and Fading Channels are some of the exciting topics covered in this chapter.

Part IV: Coding for Secure Communications

This final part consists of two practical chapters that discuss two very different approaches to secure communications.

Chapter 9 is all about Cryptography. It talks about several interesting topics, including, Symmetric (Secret Key) Cryptography and Asymmetric (Public-Key) Cryptography. The reader gets introduced to the RSA Algorithm, Pretty Good Privacy (PGP), One-way Hashing, Elliptic Curve Cryptography, Diffie-Hellman key agreement protocol, Secure Communication using Chaos Functions, Quantum Cryptography and Biometric Encryption.

Chapter 10 is again a new chapter that deals with Physical Layer Security. It has roots in information theory, which is covered in Part I of this book. The chapter covers Shannon's notion of Security, the Wiretap Model, the Gaussian Wiretap Model and the idea of Secrecy Capacity in wireless channels. Two innovative techniques, Cooperative Jamming and Artificial Noise Forwarding, are also discussed in this chapter.

OLC Supplements

The text is supported by an exhaustive website accessible at: <http://www.mhhe.com/bose/itcc3e> with the following supplements:

- *For Instructors*
 - Solutions Manual
 - PowerPoint Lecture Slides

Acknowledgements

This edition of the book owes its existence to many wonderful people, places and things. Firstly, I would like to thank the wonderful academic environment provided by IIT Delhi and the encouragement of all my colleagues in the department of Electrical Engineering. The constructive feedback received from various readers via email, and directly at various conferences, short-term courses, quality-improvement programs (QIPs) and continuing education programs (CEPs) have been instrumental in improving the contents of this book. I wish to express my appreciation to the various members of McGraw Hill Education (India) who handled the activities related to the book very efficiently at different stages. I also acknowledge the contribution of several cups of Darjeeling tea and strong coffee that have helped me put down my thoughts coherently.

Finally, special thanks are due to my dear wife, Aloka, and wonderful daughter, Harshita, for their unconditional support and love. Thank you all!

Special thanks to the reviewers mentioned here for taking out time and providing encouraging comments and valuable suggestions regarding improvement of the book.

Varunika Arya *Maharishi Markandeshwar University, Ambala, Haryana*

Amit Sehgal *G.L. Bajaj Institute of Technology & Management, Greater Noida, Uttar Pradesh*

Ashish Goel *Jaypee Institute of Information Technology, Noida*

Rubal Chaudhary *JMIT, Radaur, Haryana*

Abhijit Chandra	<i>Jadavpur University, Kolkata, West Bengal</i>
Barnali Dey	<i>Sikkim Manipal Institute of Technology, Sikkim</i>
Pinaki Mukherjee	<i>Jalpaiguri Government Engineering College, Jalpaiguri, West Bengal</i>
Tania Das	<i>Heritage Institute of Technology, Kolkata, West Bengal</i>
S P Mohani	<i>College of Engineering, Pune, Maharashtra</i>
Bhalchandra M Hardas	<i>Shri Ramdeobaba College of Engineering and Management, Nagpur, Maharashtra</i>
M Mani Roja	<i>Thadomal Shahani Engineering College, Mumbai, Maharashtra</i>
Saurabh N. Mehta	<i>Vidyalankar Institute of Technology, Mumbai, Maharashtra</i>
Hitendra Ekanath	<i>R.C. Patel Institute of Technology, Shirpur, Maharashtra</i>
Surya Vanshi	
S Radhakrishnan	<i>Kamaraj College of Engineering & Technology, Virodhunagar, Tamil Nadu</i>
A R Kavitha	<i>Jerusalem College of Engineering, Chennai, Tamil Nadu</i>
S Selvakanmani	<i>Velammal Institute of Technology, Chennai, Tamil Nadu</i>
Suresh Babu P	<i>Velammal College of Engineering and Technology, Madurai, Tamil Nadu</i>
Nishanth N	<i>Thangal Kunju Musaliar College of Engineering, Kollam, Kerala</i>
Girish N	<i>Vemana Institute of Technology, Bengaluru, Karnataka</i>

Finally, I sincerely hope that this book not only ‘covers’ all the important topics, but also ‘uncovers’ them for the students! Suggestions, comments and ideas for the next edition may please be emailed to the author at rbose@ee.iitd.ac.in.

Publisher’s Note

McGraw Hill Education (India) invites suggestions and comments from you, all of which can be sent to info.india@mheducation.com (kindly mention the title and author name in the subject line).

Piracy-related issues may also be reported.

Preface to the First Edition

Information theory, error control coding and cryptography are the three load-bearing pillars of modern digital communication systems. All the three topics are vast, and there are many good books that deal with these topics individually. In this book, an attempt has been made to incorporate all the important concepts of *information theory, error control coding and cryptography* in-between the two covers, without making the covers too far apart. This is intended as a simple and lively book on the subject.

This book results from my teaching of different topics on information theory and coding at Indian Institute of Technology, Delhi. While writing this book, I had to take a decision regarding how mathematical the book should be. Quoting Richard W. Hamming: “*Mathematics is an interesting intellectual sport but it should not be allowed to stand in the way of obtaining sensible information about physical processes*”. Too mathematical a book has the potential danger of scaring away students who lack a strong background in mathematics. On the other hand, the use of mathematics cannot be reduced beyond a limit, if the concepts in information theory and error control coding have to be studied with a certain amount of rigor. But then, life is all about striking a balance. I have tried to traverse the path of golden mean in this book. Mathematics has been used wherever necessary, and only to the extent that it is essential. Intuitive explanations have been provided wherever possible. I also believe that teaching by example is a very effective method of instruction. Therefore, as soon as a new concept is introduced, I have tried to provide at least one numerical example.

How to Read This Book

This book has been written to be both a lively introduction as well as a fairly detailed reference to the fascinating world of information theory, coding and cryptography. The entire book has been divided into three logical parts:

Part I—Information Theory and Source Coding,

Part II—Error Control Coding (Channel Coding), and

Part III—Coding for Secure Communications.

Part I contains two chapters—Chapter 1 deals with the concept of information and its efficient representation. Efficient representation of information leads to data compression. The chapter also introduces the concept of run-length coding, the rate distortion function and the design of an optimal quantizer. The chapter concludes by giving a brief introduction to image compression.

Chapter 2 deals with the concepts of a communication channel and the channel capacity. This chapter tries to answer the question: How many bits per second can be sent over a channel of a given bandwidth and for a given signal to noise ratio? It also brings out the need for error control coding.

Part II contains five chapters, all on error control coding—Chapter 3 introduces the reader to the class of linear block codes. Linear block codes are useful, instructive and simple. Encoding and decoding strategies are discussed for this class of codes. The notions of perfect codes, optimal linear codes and maximum distance separable (MDS) codes are also introduced.

Chapter 4 deals with cyclic codes, a sub-class of linear block codes. Cyclic codes are particularly useful for burst error correction. Fire codes, Golay codes and Cyclic Redundancy Check (CRC) codes are discussed as specific examples of cyclic codes. The chapter concludes with a section on the circuit implementation of cyclic codes.

Chapter 5 takes the reader to the world of Bose–Chaudhuri Hocquenghem (BCH) codes, a very powerful class of multiple error correcting codes. The Reed–Solomon codes, a sub-class of BCH codes, is also discussed in this chapter.

Chapter 6 takes a look at convolutional codes, which are essentially codes with memory. The concept of Trellis codes is introduced to the reader and the Viterbi decoding technique is discussed in detail. Some known good convolutional codes are also studied. Finally, the reader is given a flavor of the not-so-old Turbo codes.

Chapter 7 on Trellis Coded Modulation (TCM) talks about the combined coding and modulation schemes. TCM encoding and decoding are discussed. The reader also learns how to design TCM schemes for additive white Gaussian noise channels as well as fading channels.

Part III contains a chapter on Cryptography—Chapter 8 looks at yet another application of coding, i.e. secure communications. The chapter discusses both the secret key and public key encryption techniques using specific examples. Other techniques such as one-way hashing and encryption using chaos functions are also discussed. The chapter concludes with a note on the politics of cryptography.

I have tried to include numerical examples as and when required. Each chapter ends with a concluding remark, which contains brief historical notes describing the origins of the important results and contributions. Also, there is a telegraphic summary at the end of each chapter, which can be used as a ready reference, or a quick search mechanism for a particular formula or definition, or simply a confidence builder prior to an exam. The end of the chapter problems should help the enthusiastic reader crystallize the concepts discussed in the text. I have also added computer-based exercises at the end of each chapter. It is recommended that these computer problems should be made a part of learning this subject.

I have tried my best to free the book from all errors. Unfortunately, there does not exist a foolproof error control technique for that! I have tried to include all the important, practical and interesting concepts related to this area. Comments from the readers regarding errors, omissions and other constructive suggestions are welcome at rbose@ee.iitd.ac.in

Finally, I would like to quote Blaise Pascal, who said, “*The last thing one knows when writing a book is what to put first*”.

RANJAN BOSE
New Delhi

List of Abbreviations

3GPP Third Generation Partnership Project
AES Advanced Encryption Standard
AF Amplify-and-Forward
ARQ Automatic Repeat Request
ASIC Application Specific Integrated Circuit
AWGN Additive White Gaussian Noise
BCH Bose–Chaudhuri Hocquenghem
BCJR Bahl, Cocke, Jelinek and Raviv
BEC Binary Erasure Channel
BER Bit Error Rate
BPSK Binary Phase–Shift Keying
BSC Binary Symmetric Channel
CDMA Code Division Multiple Access
CIRC Cross–Interleaved Reed–Solomon Code
CRC Cyclic Redundancy Check
CSI Channel State Information
DCT Discrete Cosine Transform
DSA Digital Signature Algorithm
DEA Data Encryption Algorithm
DES Data Encryption Standard
DF Decode-and-Forward
DMC Discrete Memoryless Channel
DMS Discrete Memoryless Source
DSA Digital Signature Algorithm
DSS Digital Signature Standard
DVB Digital Video Broadcast

DVB-RCS Digital Video Broadcasting–Return Channel via Satellite
DWTC Degraded Wiretap Channel
ECC Elliptic Curve Cryptography
ECMA European Computer Manufacturers Association
FCS Frame Check Sequence
FEC Forward Error Correction
FIPS Federal Information Processing Standard
FIR Finite Impulse Response
FLC Fixed Length Code
FPGA Field Programmable Gate Array
GCD Greatest Common Divisor
GSM Global System for Mobile
GF Galois Field
GIF Graphics Interchange Format
HCCC Hybrid Concatenated Convolutional Codes
HDLC High-Level Data Link Control
HDTV High Definition Television
IDEA International Data Encryption Algorithm
IDS Iterative Decoding Suitability
ISBN International Standard Book Number
ISO International Standards Organization
JPEG Joint Photographic Experts Group
LDPC Low Density Parity Check
LTE Long Term Evolution

LZW Lempel–Ziv–Welch
MAC Message Authentication Code
MAP Maximum a-posteriori
MDS Maximum Distance Separable
MIMO Multiple Input Multiple Output
MISO Multiple Input Single Output
ML Maximum Likelihood
MPEG Moving Picture Coding Exports Group
MPSK Multi–Phase Shift Keying
MRO Mars Reconnaissance Orbiter
MSB Most Significant Bits
NSB National Bureau of Standards
PAM Pulse Amplitude Modulation
PCCC Parallel Concatenated Convolutional Codes
PEP Pairwise Error Probability
PGP Pretty Good Privacy
PKCS Public–Key Cryptography Standards
PN Pseudo–random Noise
PRGA Pseudo-random Generation Algorithm
PSK Phase Shift Keying
QAM Quadrature Amplitude Modulation
QPSK Quaternary Phase–Shift Keying

RC Rivest Cipher
RLE Run–Length Encoding
RS Reed–Solomon
RSA Rivest, Shamir and Adleman
RSC Recursive Systematic Convolutional
SCCC Serial Concatenated Convolutional Codes
SEP Symbol Error Probability
SHA Secure Hash Algorithm
SIMO Single Input Multiple Output
SISO Single Input Single Output
SNR Signal to Noise Ratio
SOVA Soft Output Viterbi Algorithm
STBC Space–Time Block Codes
STTC Space–Time Trellis Codes
TCC Turbo Convolutional Codes
TCM Trellis Coded Modulation
TPC Turbo Product Code
UWB Ultra Wideband
VCEG Video Coding Experts Group
VLC Variable Length Code
VSAT Very Small Aperture Terminal
WTC Wiretap Channel

Pre-requisite

Start by doing what's necessary; then do what's possible; and suddenly you are doing the impossible.

Francis of Assisi (1182-1226)

0.1 Introduction to Matrices

Definition 1 A **matrix** is a rectangular array of elements which can be real or complex

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MN} \end{bmatrix} \quad (1)$$

The element in the i^{th} row and j^{th} column is written as a_{ij} .

Definition 2 A matrix with only one column is called a **column vector** or just a **vector**. The number of rows of a vector is called its dimension. For example, an N -dimensional vector \mathbf{x} is given by

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} \quad (2)$$

Definition 3 The **Euclidean norm** of an N -dimensional row vector or vector, also called its **norm**, is defined as

$$\|\mathbf{x}\| = \sqrt{\sum_{i=1}^N |x_i|^2} \quad (3)$$

Definition 4 The sum of two matrices, \mathbf{A} and \mathbf{B} , is given by $\mathbf{C} = \mathbf{A} + \mathbf{B}$, where

$$c_{ij} = a_{ij} + b_{ij} \quad (4)$$

Definition 5 The product of two matrices, \mathbf{A} and \mathbf{B} , is given by $\mathbf{C} = \mathbf{AB}$, where

$$c_{ij} = \sum_k a_{ik} b_{kj} \quad (5)$$

Note:

- (a) Matrix addition is associative, i.e., $\mathbf{A} + (\mathbf{B} + \mathbf{C}) = (\mathbf{A} + \mathbf{B}) + \mathbf{C}$.
- (b) Matrix addition is commutative, i.e., $(\mathbf{A} + \mathbf{B}) = (\mathbf{B} + \mathbf{A})$.
- (c) Matrix multiplication is associative, i.e., $\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C}$.
- (d) In general, matrix multiplication is non-commutative, i.e., $\mathbf{AB} \neq \mathbf{BA}$.

Definition 6 The identity matrix is given by

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \quad (6)$$

Definition 7 Let \mathbf{A} be an $n \times m$ matrix. The **transpose** of \mathbf{A} , denoted by \mathbf{A}^T , is an $m \times n$ matrix formed by interchanging the rows and columns of \mathbf{A} .

$$\mathbf{B} = \mathbf{A}^T \text{ implies } b_{ij} = a_{ji} \quad (7)$$

Note:

- (a) $(\mathbf{A}^T)^T = \mathbf{A}$
- (b) $(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T$
- (c) $(\alpha\mathbf{A})^T = \alpha\mathbf{A}^T$
- (d) $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$

Definition 8 A matrix \mathbf{A} is said to be **invertible**, if there exists a matrix \mathbf{B} such that

$$\mathbf{BA} = \mathbf{AB} = \mathbf{I} \quad (8)$$

The inverse of a square matrix \mathbf{A} is denoted by \mathbf{A}^{-1} , where

$$\mathbf{AA}^{-1} = \mathbf{I} = \mathbf{A}^{-1}\mathbf{A} \quad (9)$$

Definition 9 The number of linearly independent column vectors in a matrix A is called the column rank of A , and the number of linearly independent row vectors in a matrix A is called the row rank of A . It can be shown that Rank of A = column rank of A = row rank of A .

Note:

- (a) If the rank of an $N \times N$ matrix A is N , then A is **invertible** (or **nonsingular**) and A^{-1} exists.
- (b) If a matrix is not invertible, we say it is **singular** or **noninvertible**.
- (c) Not every square matrix is invertible.
- (d) $(A^{-1})^{-1} = A$.
- (e) $(AB)^{-1} = B^{-1}A^{-1}$
- (f) $A^{-k} = (A^{-1})^k$

Definition 10 Let A be an $N \times N$ matrix. If

$$Ax = \lambda x \quad (10)$$

for some scalar λ and nonzero column vector x , then A is called an **eigenvalue** (or characteristic value) of A and x is called an **eigenvector** associated with λ .

Definition 11 The **complex conjugate** A^* of a matrix A is obtained by taking the complex conjugate of each element of A , as given

$$A^* = \begin{bmatrix} a_{11}^* & a_{12}^* & \cdots & a_{1N}^* \\ a_{21}^* & a_{22}^* & \cdots & a_{2N}^* \\ \vdots & \vdots & \ddots & \vdots \\ a_{M1}^* & a_{M2}^* & \cdots & a_{MN}^* \end{bmatrix} \quad (11)$$

Definition 12 **Hermitian** of a matrix A is defined as its conjugate transpose, i.e.,

$$A^H = (A^*)^T \quad (12)$$

Definition 13 A **Hermitian matrix** A (observe the *italicised* difference with the previous definition) is a square matrix with complex entries that is equal to its own conjugate transpose, i.e.,

$$A = (A^*)^T = A^H \quad (13)$$

where A^H is the conjugate transpose of A .

Note:

- (a) A Hermitian matrix is also called a **self-adjoint matrix**.
- (b) All eigenvalues of a Hermitian matrix are real, although the eigenvectors can be complex.

Definition 14 A matrix \mathbf{U} is said to be **unitary**, if

$$\mathbf{U}^{-1} = \mathbf{U}^H \quad \text{or} \quad \mathbf{U}^H \mathbf{U} = \mathbf{I} \quad (14)$$

Note:

- (a) The columns of \mathbf{U} form an orthogonal set.
- (b) The rows of \mathbf{U} form an orthogonal set.
- (c) If \mathbf{U} is unitary then \mathbf{U}^H is unitary.
- (d) All eigenvalues of a unitary matrix have absolute value 1.

Definition 15 A matrix \mathbf{A} is said to be **normal** if

$$\mathbf{A}^H \mathbf{A} = \mathbf{A} \mathbf{A}^H \quad (15)$$

Note:

- (a) All unitary matrices are normal.
- (b) All Hermitian matrices are normal.
- (c) Every normal matrix can be written as $\mathbf{A} = \mathbf{U} \Lambda \mathbf{U}^H$ where \mathbf{U} is unitary and Λ is a diagonal matrix whose diagonal entries are the eigenvalues of \mathbf{A} .

Definition 16 An **orthogonal matrix** \mathbf{A} has the property

$$\mathbf{A} \mathbf{A}^T = \mathbf{I} \quad (16)$$

where \mathbf{A}^T is the transpose of \mathbf{A} .

Definition 17 A **symmetric matrix** \mathbf{A} has the property

$$\mathbf{A} = \mathbf{A}^T \quad (17)$$

where \mathbf{A}^T is the transpose of \mathbf{A} .

Definition 18 **Singular Value Decomposition** implies that for any matrix \mathbf{A}

$$\mathbf{A} = \mathbf{V} \Sigma \mathbf{W}^H \quad (18)$$

where \mathbf{V} and \mathbf{W} are unitary and Σ is a diagonal matrix with non-negative main diagonal entries. The rank of Σ is the same as that of \mathbf{A} .

0.2 Introduction to Probability Theory

Definition 19 We typically represent the outcome of an experiment by a capital Roman letter, such as X , called a **random variable**. The sample space of the experiment is the set of all possible outcomes. If the **sample space** is either finite or countably infinite, the random variable is said to be discrete.

Definition 20 A **distribution function** for X is a real-valued function m whose domain is Ω and which satisfies:

- (a) $m(\sigma) \geq 0$, for all $\sigma \in \Omega$ and
- (b) $\sum_{\sigma \in \Omega} m(\sigma) = 1$.

For any subset E of Ω , we define the probability of E to be the number $P(E)$ given by

$$P(E) = \sum_{\sigma \in E} m(\sigma) \quad (19)$$

Note:

- (a) $P(E) \geq 0$ for every $E \subset \Omega$.
- (b) $P(\Omega) = 1$.
- (c) If $E \subset F \subset \Omega$, then $P(E) \leq P(F)$.
- (d) If A and B are disjoint subsets of Ω , then $P(A \cup B) = P(A) + P(B)$.
- (e) If A and B are subsets of Ω , $P(A \cup B) = P(A) + P(B) - P(A \cap B)$.

Definition 21 Let X be a **continuous real-valued random variable**. Then the **cumulative distribution function** of X is defined by the equation

$$F_X(x) = P(X \leq x) \quad (20)$$

Definition 22 Let X be a continuous real-valued random variable with **probability density function** $f(x)$. Then the function is defined by

$$F(x) = \int_{-\infty}^x f(t)dt \quad (21)$$

which is the **cumulative distribution function** of X . Furthermore,

$$\frac{d}{dx} F(x) = f(x) \quad (22)$$

Since $F(\infty) = 1$ and $F(-\infty) = 0$, consequently,

$$\int_{-\infty}^{\infty} f(x)dx = 1 \quad (23)$$

Definition 23 We call $P(F|E)$ the conditional probability of F occurring given that E occurs, and compute it using

$$P(F | E) = \frac{P(F \cap E)}{P(E)} \quad (24)$$

Definition 24 Suppose we have a set of events H_1, H_2, \dots, H_m that are pairwise disjoint and such that the sample space satisfies $\Omega = H_1 \cup H_2 \cup \dots \cup H_m$ then the **Bayes' formula** states that

$$P(H_i | E) = \frac{P(H_i)P(E | H_i)}{\sum_{k=1}^m P(H_k)P(E | H_k)} \quad (25)$$

Definition 25 Let X_1, X_2, \dots, X_n be continuous random variables associated with an experiment, and let $\tilde{X} = (X_1, X_2, \dots, X_n)$. Then the **joint cumulative distribution function** of \tilde{X} is defined by

$$F(x_1, x_2, \dots, x_n) = P(X_1 \leq x_1, X_2 \leq x_2, \dots, X_n \leq x_n) \quad (26)$$

Definition 26 Let X_1, X_2, \dots, X_n be continuous random variables with cumulative distribution functions $F_1(x), F_2(x), \dots, F_n(x)$ and with density functions $f_1(x), f_2(x), \dots, f_n(x)$ respectively. Then, these random variables are **mutually independent** if

$$F(x_1, x_2, \dots, x_n) = F_1(x_1)F_2(x_2) \dots F_n(x_n) \quad (27)$$

or

$$f(x_1, x_2, \dots, x_n) = f_1(x_1)f_2(x_2) \dots f_n(x_n) \quad (28)$$

for any choice of x_1, x_2, \dots, x_n .

Definition 27 Let X be a discrete random variable with sample space Ω and distribution function $f(x)$. The **expected value** is defined as

$$E(X) = \sum_{x \in \Omega} xf(x) \quad (29)$$

provided this sum converges absolutely. We often refer to the expected value as the **mean**.

Let X and Y be random variables with finite expected values. Then $E(X + Y) = E(X) + E(Y)$.

Definition 28 The **mean** or **expected value** of a continuous random variable X is its probabilistic average, defined as

$$\mu = E[X] = \int_{-\infty}^{\infty} xf(x)dx \quad (30)$$

The expectation operator $E[]$ is **linear** and can also be applied to functions of random variables given as follows:

$$E[g(X)] = \int_{-\infty}^{\infty} g(x)f(x)dx \quad (31)$$

Definition 29 The n^{th} moment of X is defined as

$$E[X^n] = \int_{-\infty}^{\infty} x^n f(x)dx \quad (32)$$

Definition 30 The variance of X is defined as

$$\text{Var}[X] = E[(X - \mu)^2] = E[X^2] - (\mu)^2 \quad (33)$$

Definition 31 The **uniform distribution** on a sample space Ω containing n elements is defined by

$$P(\sigma_i) = \frac{1}{n} \text{ for every } \sigma_i \in \Omega \quad (34)$$

Definition 32 The **binomial distribution** is given by

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k} \quad (35)$$

where $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ and $n!$ represents factorial n .

Note:

- (a) **Stirling's Formula:** Gives the asymptotic value of factorial n

$$n! \approx n^n e^{-n} \sqrt{2\pi n} \quad (36)$$

- (b) **Binomial coefficients:** For integers n and j , with $0 < j < n$, the binomial coefficients satisfy:

$$\binom{n}{j} = \binom{n-1}{j} + \binom{n-1}{j-1} \quad (37)$$

Definition 33 The **Poisson distribution** is given by

$$P(X = k) = \frac{\lambda^k}{k!} e^{-\lambda} \quad (38)$$

Definition 34 The **continuous uniform distribution** over the interval $[a, b]$ is given by

$$f(x) = \begin{cases} \frac{1}{b-a}, & \text{if } a \leq x \leq b \\ 0, & \text{otherwise} \end{cases} \quad (39)$$

Definition 35 The **exponential distribution** is given by

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & \text{if } 0 \leq x < \infty \\ 0, & \text{otherwise} \end{cases} \quad (40)$$

where λ is a positive constant.

Definition 36 The **Gaussian distribution** is given by

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2} \quad (41)$$

where μ is the mean and σ^2 is the variance. The Gaussian distribution, also called the **normal distribution**, is denoted as $N(\mu, \sigma^2)$.

Definition 37 The **Rayleigh distribution** is given by

$$f(x) = \frac{x}{\sigma^2} e^{-x^2/2\sigma^2} \quad (42)$$

where $x \geq 0$.

Definition 38 The **Central Limit Theorem** states that if we sum a large number of mutually independent random variables, then the distribution of the sum can be closely approximated by the normal density.

Live as if you were to die tomorrow. Learn as if you were to live forever.

Mahatma Gandhi (1869 – 1948)

PART - I

Information Theory and Source Coding

Chapter 1 Source Coding

Chapter 2 Channel Capacity and Coding

Source Coding

Not everything that can be counted counts, and not everything that counts can be counted.

Albert Einstein (1879-1955)

learning objectives

After studying this chapter the students will be able to:

- LO 1** Understanding the concept of information.
- LO 2** State and prove the source coding theorem and discuss its ramifications.
- LO 3** Analyze five source coding techniques: the Huffman encoding, Shannon-Fano-Elias encoding, Arithmetic encoding, the Lempel-Ziv encoding and Run Length encoding.
- LO 4** Underline the concept of the rate distortion function and the design of the optimum quantizer.
- LO 5** Apply the knowledge to study image compression, one of the important application areas of source coding.

1.1 Introduction to Information Theory

Today we live in the information age. Internet has become an integral part of our lives, making this third planet from the sun, a global village. People are seen talking over the cellular phones, sometimes even in cinema halls. Movies can be rented in the form of a DVD disk or streamed directly to our smart phones. Email and web addresses are commonly visible on business cards. Most of the people today prefer to send emails and e-cards to their friends rather than the regular snail mail. Stock quotes and cricket scores can be checked using the mobile phone. ‘Selfies’ can be captured and uploaded on social media sites with just a click of a button.

...
This chapter comes with a video
overview by the author. Scan here to
know more or

: Visit <http://qrcode.flipick.com/index.php/582>



DID YOU KNOW ? Information has become a key to success (it has always been the key to success, but in today's world it is *the* key). And behind all this exchange of information lie the tiny 1's and 0's (the omnipresent bits) that hold the information by the mere way they sit next to one another. Yet the present day's information age owes primarily to a seminal paper published in 1948 that laid the foundation of the wonderful field of **Information Theory**—a theory that was initiated by one man, the American Electrical Engineer Claude E. Shannon, whose ideas appeared in the article "*The Mathematical Theory of Communication*" in the *Bell System Technical Journal* (1948). In its broadest sense, information is interpreted to include the messages occurring in any of the standard communications media, such as telephone, radio, or television, and the signals involved in electronic computers, electromechanical systems, and other data-processing devices. The theory is even applicable to the signals appearing in the nerve networks of humans and other animals.

The chief concern of information theory is to discover mathematical laws governing systems designed to communicate or manipulate information. It sets up quantitative measures of information and of the capacity of various systems to transmit, store, and otherwise process information. Some of the problems treated are related to finding the best methods of using various available communication systems and the best methods for separating the wanted information, or signal, from the extraneous information, or noise. Another problem is the setting of upper bounds on what it is possible to achieve with a given information-carrying medium (often called an information channel). While the results are chiefly of the interest to communication engineers, some of the concepts have been adopted and found useful in fields like psychology and linguistics. The notion of mutual information has also found applications in population based gene mapping, whose aim is to find DNA regions (genotypes) responsible for particular traits (phenotypes).



The boundaries of information theory are quite fuzzy. The theory overlaps heavily with communication theory but is more oriented toward the fundamental limitations on the processing and communication of information and less oriented toward the detailed operation of the devices employed.

In this chapter

We will start with an intuitive understanding of information and link uncertainty to information. We will then define self-information, average self-information (entropy), mutual information, average mutual information and relative entropy, in LO 1.

Next, we shall state and prove the source coding theorem and also look at some examples of variable and fixed length codes. We will also introduce the notion of a prefix code, in LO 2.

We will then study five interesting source coding techniques: the Huffman encoding, Shannon-Fano-Elias encoding, Arithmetic encoding, Lempel-Ziv encoding and the Run Length encoding, in LO 3.

Next, we will understand the concept of the rate distortion function and the design of the optimum quantizer. We will find out how to calculate the entropy rate for a stochastic process. We will also study the Markov process, in LO 4.

Finally, we will use all the beautiful concepts that we have learnt and apply the knowledge to study image compression, one of the important application areas of source coding. In particular, the JPEG compression standard will be discussed, in LO 5.

1.2 Uncertainty and Information

Any information source produces an output that is random in nature. If the source output had no randomness, *i.e.*, the output were known exactly, there would be no need to transmit it! There exist both **analog** and **discrete** information sources. Actually, we live in an analog world, and most sources are analog sources, for example, speech, temperature fluctuations, etc. The discrete sources are man made sources, for example, a source (say, a man) that generates a sequence of letters from a finite alphabet (while typing email).

Before we go on to develop a mathematical measure of information, let us develop an intuitive feel for it. Read the following sentences:

- (A) Tomorrow, the sun will rise from the East.
- (B) The phone will ring in the next one hour.
- (C) It will snow in Delhi this winter.



Intuition The three sentences carry different amounts of information. In fact, the first sentence hardly carries any information. It is a sure-shot thing. Everybody knows that the sun rises from the East and the probability of this happening again is almost unity (“*Making predictions is risky, especially when it involves the future.*” - N. Bohr). Sentence (B) appears to carry more information than sentence (A). The phone may ring, or it may not. There is a finite probability that the phone will ring in the next one hour (unless the maintenance people are at work again). The last sentence probably made you read it over twice. This is because it has never snowed in Delhi, and the probability of a snowfall is very low. It is interesting to note that the amount of information carried by the sentences listed above have something to do with the probability of occurrence of the events stated in the sentences. And we observe an inverse relationship. Sentence (A), which talks about an event which has a probability of occurrence very close to 1 carries almost no information. Sentence (C), which has a very low probability of occurrence, appears to carry a lot of information (made us read it twice to be sure we got the information right!). The other interesting thing to note is that the length of the sentence has nothing to do with the amount of information it conveys. In fact, sentence (A) is the longest of the three sentences but carries the minimum information.

We will now develop a mathematical measure of information.

Definition 1.1 Consider a discrete random variable X with possible outcomes x_i , $i = 1, 2, \dots, n$. The **Self Information** of the event $X = x_i$ is defined as

$$I(x_i) = \log\left(\frac{1}{P(x_i)}\right) = -\log P(x_i) \quad (1.1)$$

We note that a high probability event conveys less information than a low probability event. For an event with $P(x_i) = 1$, $I(x_i) = 0$. Since a lower probability implies a higher degree of uncertainty (*and vice versa*), a random variable with a higher degree of uncertainty contains more information. We will use this correlation between uncertainty and information for physical interpretations throughout this chapter.

LO 1



Understanding the concept of information.



The units of $I(x_i)$ are determined by the base of the logarithm, which is usually selected as 2 or e . When the base is 2, the units are in **bits** and when the base is e , the units are in **nats** (natural units). Since $0 \leq P(x_i) \leq 1$, $I(x_i) \geq 0$, i.e., self-information is *non-negative*. The following two examples illustrate why a logarithmic measure of information is appropriate.

Example 1.1 Consider a binary source which tosses a fair coin and produces an output equal to

1 if a head appears and a 0 if a tail appears. For this source, $P(1) = P(0) = 0.5$. The information content of each output from the source is

$$I(x_i) = -\log_2 P(x_i) = -\log_2(0.5) = 1 \text{ bit} \quad (1.2)$$

Indeed, we have to use only one bit to represent the output from this binary source (say, we use a 1 to represent H and a 0 to represent T). Now, suppose the successive outputs from this binary source are statistically independent, i.e., the source is memoryless. Consider a block of m binary digits. There are 2^m possible m -bit blocks, each of which is equally probable with probability 2^{-m} . The self information of an m -bit block is

$$I(x_i) = -\log_2 P(x_i) = -\log_2 2^{-m} = m \text{ bits} \quad (1.3)$$

Again, we observe that we indeed need m bits to represent the possible m -bit blocks. Thus, this logarithmic measure of information possesses the desired additive property when a number of source outputs are considered as a block.

Example 1.2 Consider a discrete, memoryless source (source C) that generates *two* bits at a time. This source comprises two binary sources (sources A and B) as mentioned in Example 1.1, each source contributing one bit. The two binary sources within the source C are independent. Intuitively, the information content of the aggregate source (source C) should be the *sum* of the information contained in the outputs of the two independent sources that constitute this source C . Let us look at the information content of the outputs of source C . There are four possible outcomes {00, 01, 10, 11}, each with a probability $P(C) = P(A)P(B) = (0.5)(0.5) = 0.25$, because the sources A and B are independent. The information content of each output from the source C is

$$\begin{aligned} I(C) &= -\log_2 P(x_i) \\ &= -\log_2(0.25) = 2 \text{ bits} \end{aligned} \quad (1.4)$$

We have to use two bits to represent the output from this combined binary source. Thus, the logarithmic measure of information possesses the desired additive property for independent events.

Next, consider *two* discrete random variables X and Y with possible outcomes x_i , $i = 1, 2, \dots, n$. and y_j , $j = 1, 2, \dots, m$ respectively. Suppose we observe some outcome $Y = y_j$ and we want to determine the amount of information this event provides about the event $X = x_i$, $i = 1, 2, \dots, n$, i.e., we want to mathematically represent the mutual information. We note the two extreme cases:



- (i) X and Y are independent, in which case the occurrence of $Y = y_j$ provides no information about $X = x_i$.
- (ii) X and Y are fully dependent events, in which case the occurrence of $Y = y_j$ determines the occurrence of the event $X = x_i$.

A suitable measure that satisfies these conditions is the logarithm of the ratio of the conditional probability

$$P(X = x_i | Y = y_j) = P(x_i | y_j) \quad (1.5)$$

divided by the probability

$$P(X = x_i) = P(x_i). \quad (1.6)$$

Definition 1.2 The Mutual Information $I(x_i; y_j)$ between x_i and y_j is defined as

$$I(x_i; y_j) = \log\left(\frac{P(x_i|y_j)}{P(x_i)}\right) \quad (1.7)$$

As before, the units of $I(x_i; y_j)$ are determined by the base of the logarithm, which is usually selected as 2 or e . When the base is 2 the units are in bits. Note that

$$\frac{P(x_i|y_j)}{P(x_i)} = \frac{P(x_i|y_j)P(y_j)}{P(x_i)P(y_j)} = \frac{P(x_i, y_j)}{P(x_i)P(y_j)} = \frac{P(y_j|x_i)}{P(y_j)} \quad (1.8)$$

Therefore,

$$I(x_i; y_j) = \log\left(\frac{P(x_i|y_j)}{P(x_i)}\right) = \log\left(\frac{P(y_j|x_i)}{P(y_j)}\right) = I(y_j; x_i) \quad (1.9)$$

The physical interpretation of $I(x_i; y_j) = I(y_j; x_i)$ is as follows:

The information provided by the occurrence of the event $Y = y_j$ about the event $X = x_i$ is identical to the information provided by the occurrence of the event $X = x_i$ about the event $Y = y_j$.

Let us now verify the two extreme cases:

- (i) When the random variables X and Y are statistically independent, $P(x_i | y_j) = P(x_i)$, which leads to $I(x_i; y_j) = 0$.
- (ii) When the occurrence of $Y = y_j$ uniquely determines the occurrence of the event $X = x_i$, $P(x_i | y_j) = 1$, and the mutual information becomes

$$I(x_i; y_j) = \log\left(\frac{1}{P(x_i)}\right) = -\log P(x_i) \quad (1.10)$$

This is the self information of the event $X = x_i$.

Thus, the logarithmic definition of mutual information confirms our intuition.

Example 1.3 Consider a binary symmetric channel (BSC) as shown in Fig. 1.1. A binary channel can be visualized as one that transports 1's and 0's from the transmitter (Tx) to the receiver (Rx). It makes an error occasionally, with probability p . A BSC channel flips a 1 to 0 and *vice-versa* with equal probability. Let X and Y be binary random variables that represent the input and output of this BSC. Let the input symbols be equally likely, and the output symbols depend upon the input according to the channel transition probabilities are as follows:

$$P(Y=0 | X=0) = 1-p.$$

$$P(Y=0 | X=1) = p.$$

$$P(Y=1 | X=1) = 1-p.$$

$$P(Y=1 | X=0) = p.$$

It simply implies that the probability of a bit getting flipped (in error) when transmitted over this BSC is p . From the channel transition probabilities we have

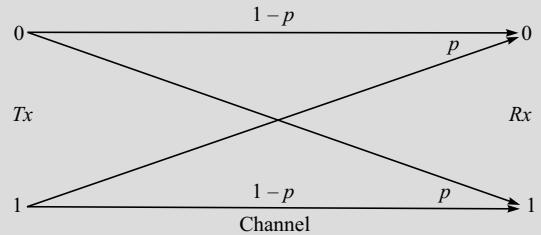


Fig. 1.1 A binary symmetric channel.

$$\begin{aligned} P(Y=0) &= P(X=0).P(Y=0 | X=0) + P(X=1).P(Y=0 | X=1) \\ &= 0.5(1-p) + 0.5(p) = 0.5 \end{aligned}$$

and

$$\begin{aligned} P(Y=1) &= P(X=0).P(Y=1 | X=0) + P(X=1).P(Y=1 | X=1) \\ &= 0.5(p) + 0.5(1-p) = 0.5 \end{aligned}$$

Let us suppose that we are at the receiver end and we want to comment about what was transmitted at the transmitter, having observed what is received at the receiver end. The mutual information about the occurrence of the event $X=0$ given that $Y=0$ is observed is

$$I(x_0; y_0) = I(0; 0) = \log_2 \left(\frac{P(Y=0 | X=0)}{P(Y=0)} \right) = \log_2 \left(\frac{1-p}{0.5} \right) = \log_2 2(1-p).$$

$$\text{Similarly, } I(x_1; y_0) = I(1; 0) = \log_2 \left(\frac{P(Y=0 | X=1)}{P(Y=0)} \right) = \log_2 \left(\frac{p}{0.5} \right) = \log_2 2p.$$

Let us consider some specific cases. Suppose, $p = 0$, i.e., it is an ideal channel (noiseless). In that case

$$I(x_0; y_0) = I(0; 0) = \log_2 2(1-p) = 1 \text{ bit.}$$

Hence having observed the output we can determine what was transmitted with certainty. Recall that the self-information about the event $X=x_0$ was 1 bit. However, if $p = 0.5$, we obtain

$$I(x_0; y_0) = I(0; 0) = \log_2 2(1-p) = \log_2 2(0.5) = 0.$$



Interpretation

This implies that having observed the output, we have no information about what was transmitted. Thus, it is a useless channel. For such a channel, there is no point in observing the received symbol and trying to make a guess as to what was sent. Instead, we can also toss a fair coin at the receiver in order to estimate what was sent!

Suppose we have a channel where $p = 0.1$. For this channel,

$$I(x_0; y_0) = I(0; 0) = \log_2 2(1-p) = \log_2 2(0.9) = 0.848 \text{ bits.}$$

Example 1.4

Let X and Y be binary random variables that represent the input and output of a binary channel shown in Fig. 1.2. Can you think of a real-life channel that has unequal crossover probabilities for '1' and '0'? Let the input symbols be equally likely, and the output symbols depend upon the input according to the channel transition probabilities:

$$P(Y=0 | X=0) = 1-p_0.$$

$$P(Y=0 | X=1) = p_1.$$

$$P(Y=1 | X=1) = 1-p_1.$$

$$P(Y=1 | X=0) = p_0.$$

From the channel transition probabilities we have

$$\begin{aligned} P(Y=0) &= P(X=0).P(Y=0 | X=0) + P(X=1).P(Y=0 | X=1) \\ &= 0.5(1-p_0) + 0.5(p_1) = 0.5(1-p_0 + p_1) \end{aligned}$$

and

$$\begin{aligned} P(Y=1) &= P(X=0).P(Y=1 | X=0) + P(X=1).P(Y=1 | X=1) \\ &= 0.5(p_0) + 0.5(1-p_1) = 0.5(1-p_1 + p_0). \end{aligned}$$

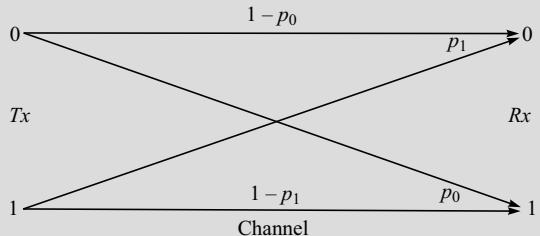


Fig. 1.2 A binary channel with asymmetric probabilities.

Suppose that we are at the receiver end and we want to comment about what was transmitted at the transmitter, having observed what is received at the receiver. The mutual information about the occurrence of the event $X = 0$ given that $Y = 0$ is observed is

$$I(x_0; y_0) = I(0; 0) = \log_2 \left(\frac{P(Y = 0 | X = 0)}{P(Y = 0)} \right) = \log_2 \left(\frac{1 - p_0}{0.5(1 - p_0 + p_1)} \right) = \log_2 \left(\frac{2(1 - p_0)}{1 - p_0 + p_1} \right)$$

Similarly, $I(x_1; y_0) = I(1; 0) = \log_2 \left(\frac{P(Y = 0 | X = 1)}{P(Y = 0)} \right) = \log_2 \left(\frac{2p_1}{1 - p_0 + p_1} \right)$

Definition 1.3 The **Conditional Self-Information** of the event $X = x_i$ given $Y = y_j$ is defined as

$$I(x_i | y_j) = \log \left(\frac{1}{P(x_i | y_j)} \right) = -\log P(x_i | y_j) \quad (1.11)$$

Thus, we may write

$$I(x_i; y_j) = I(x_i) - I(x_i | y_j) \quad (1.12)$$



The conditional self-information can be interpreted as the self-information about the event $X = x_i$, having observed the event $Y = y_j$. Recall that both $I(x_i) \geq 0$ and $I(x_i | y_j) \geq 0$. Therefore, $I(x_i; y_j) < 0$ when $I(x_i) < I(x_i | y_j)$ and $I(x_i; y_j) > 0$ when $I(x_i) > I(x_i | y_j)$. Hence, mutual information can be positive, negative or zero.

Example 1.5 Consider the BSC discussed in Example 1.3. The plot of the mutual information $I(x_0; y_0)$ versus the probability of error, p is given in Fig. 1.3.

It can be seen from Fig. 1.3 that $I(x_0; y_0)$ is negative for $p > 0.5$. The physical interpretation is as follows. A negative mutual information implies that having observed $Y = y_0$, we must *avoid* choosing $X = x_0$ as the transmitted bit.

For $p = 0.1$, $I(x_0; y_1) = I(0; 1) = \log_2 2(p) = \log_2 2(0.1) = -2.322$ bits.

This shows that the mutual information between the events $X = x_0$ and $Y = y_1$ is negative for $p = 0.1$. For the extreme case of $p = 1$, we have

$$I(x_0; y_1) = I(0; 1) = \log_2 2(p) = \log_2 2(1) = 1 \text{ bit}$$

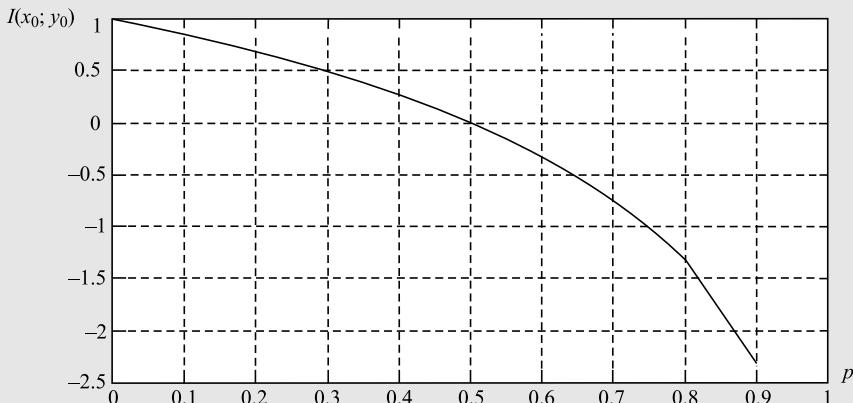


Fig. 1.3 The plot of the mutual information $I(x_0; y_0)$ versus the probability of error, p .

The channel always changes a 0 to a 1 and vice versa (since $p = 1$). This implies that if y_1 is observed at the receiver, it can be concluded that x_0 was actually transmitted. This is actually a useful channel with a 100 % bit error rate! We just flip the received bit.

1.3 Average Mutual Information and Entropy

LO 1

So far we have studied the mutual information associated with a pair of events x_i and y_j which are the possible outcomes of the two random variables X and Y . We now want to find out the average mutual information between the two random variables. This can be obtained simply by weighting $I(x_i; y_j)$ by the probability of occurrence of the joint event and summing over all possible joint events.

Definition 1.4 The **Average Mutual Information** between two random variables X and Y is given by

$$\begin{aligned} I(X; Y) &= \sum_{i=1}^n \sum_{j=1}^m P(x_i, y_j) I(x_i; y_j) = \sum_{i=1}^n \sum_{j=1}^m P(x_i, y_j) \log \frac{P(x_i, y_j)}{P(x_i)P(y_j)} \\ &= \sum_{i=1}^n \sum_{j=1}^m P(x_i)P(y_j|x_i) \log \frac{P(y_j|x_i)}{P(y_j)} \\ &= \sum_{j=1}^m \sum_{i=1}^n P(y_j)P(x_i|y_j) \log \frac{P(x_i|y_j)}{P(x_i)} \end{aligned} \quad (1.13)$$

For the case when X and Y are statistically independent, $I(X; Y) = 0$, i.e., there is no average mutual information between X and Y . An important property of the average mutual information is that $I(X; Y) \geq 0$, with equality if and only if X and Y are statistically independent.

Definition 1.5 The **Average Self Information** of a random variable X is defined as

$$H(X) = \sum_{i=1}^n P(x_i)I(x_i) = -\sum_{i=1}^n P(x_i)\log P(x_i) \quad (1.14)$$

When X represents the alphabet of possible output letters from a source, $H(X)$ represents the average information per source letter. In this case $H(X)$ is called the **entropy**. The entropy of X can be interpreted

as the expected value of $\log\left(\frac{1}{P(X)}\right)$. The term entropy has been borrowed from statistical mechanics,

where it is used to denote the level of disorder in a system. It is interesting to see that the Chinese character for entropy looks like 熵!

We observe that since $0 \leq P(x_i) \leq 1$, $\log\left(\frac{1}{P(x_i)}\right) \geq 0$. Hence, $H(X) \geq 0$.

Example 1.6 Consider a discrete binary source that emits a sequence of statistically independent symbols. The output is either 0 with probability p or 1 with a probability $1-p$. The entropy of this binary source is

$$\begin{aligned}
 H(X) &= -\sum_{i=0}^1 P(x_i) \log P(x_i) \\
 &= -p \log_2(p) - (1-p) \log_2(1-p) \quad (1.15)
 \end{aligned}$$

The plot of the *binary entropy function* versus p is given in Fig. 1.4.

We observe from the figure that the value of the binary entropy function reaches its maximum value for $p = 0.5$, i.e., when both 1 and 0 are equally likely. In general, it can be shown that the entropy of a discrete source is maximum when the letters from the source are equally probable.

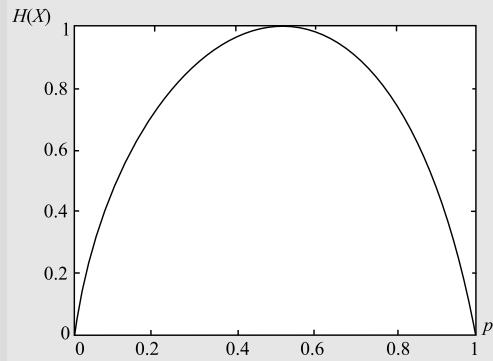


Fig. 1.4 The entropy function, $H(X) = -p \log_2(p) - (1-p) \log_2(1-p)$.

Example 1.7 Consider the English language with alphabet {A, B, ..., Z}. If every letter occurred with the same probability and was independent from the other letters, then the entropy per letter would be $\log_2 26 = 4.70$. This is the absolute upperbound. However, we know that all letters do not appear with equal probability. If we take into consideration the probabilities of occurrences of different alphabets (normalized letter frequency), the entropy per letter, H_L , would be

$$H_L \leq H(X) \approx 4.14 \text{ bits} \quad (1.16)$$

This is still an over estimate because the letters are not independent. For example Q is always followed by U and the **bigram** TH is likely to occur frequently. Thus a better estimate of the entropy per letter could be possibly obtained by looking at the bigrams. If X^2 denotes the random variable of bigrams in the English language, the upperbound on H_L can be refined as

$$H_L \leq \frac{H(X^2)}{2} \approx 3.56 \text{ bits} \quad (1.17)$$

This logic can be extended to **n-grams**. Thus the entropy of the language can be defined as

$$H_L = \lim_{n \rightarrow \infty} \frac{H(X^n)}{n} \quad (1.18)$$

Even though the exact value of H_L may be difficult to determine, statistical investigations show that for the English language

$$1 \leq H_L \leq 1.5 \text{ bits} \quad (1.19)$$

So each letter in the English text gives at most 1.5 bits of information. Let us assume the value of $H_L \approx 1.25$ bits. Thus the redundancy of the English language is

$$R_{\text{Eng}} = 1 - \frac{H_L}{\log_2 26} = 1 - \frac{1.25}{4.70} \approx 0.75 \quad (1.20)$$

This suggests that, theoretically, the English text can be compressed without loss to one-fourth of its size. It is interesting that most natural languages in the world have similar redundancies.



Let us now briefly consider the redundancy in spoken English. Suppose an average speaker speaks 60 words per minute and the average number of letters per word is 6. The average number of letters spoken per second in this case is 6 letters per second. Assuming each letter carries 1.25 bits of information, the information rate of an average speaker is 7.5 bits/second. If each letter is represented by 5 bits, the bit rate of an average speaker is 30 bits/s. However, the typical data rate requirement for speech is 32 kilobits/s!



INDUSTRIAL RELEVANCE

Low-bit-rate speech coding is used both for communication and voice storage applications. Typical rates are below 4 kb/s. At such low rates, full encoding of the speech waveform is not possible. These low-rate coders rely on parametric models to represent only the most perceptually-relevant aspects of speech. At bit rates above 4 kb/s, speech-specific waveform coders based on code excited linear prediction (CELP) is used to produce good quality speech. Low-bit-rate speech coding is used in U.S. MIL-STD 3005 2.4 kb/s MELP, the NATO STANAG4591 MELP coding suite at 2400, 1200, and 600 b/s and the ITU 4 kb/s hybrid MELP/CELP speech coder.

Definition 1.6 The Average Conditional Self Information called the **Conditional Entropy** is defined as

$$H(X|Y) = \sum_{i=1}^n \sum_{j=1}^m P(x_i, y_j) \log \frac{1}{P(x_i|y_j)} \quad (1.21)$$

The physical interpretation of this definition is as follows. $H(X|Y)$ is the information (or uncertainty) in X after Y is observed. Based on the definitions of $H(X|Y)$ and $H(X)$ we can write

$$I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \quad (1.22)$$



Interpretation

We make the following observations.

- (i) Since $I(X; Y) \geq 0$, it implies that $H(X) \geq H(X|Y)$.
- (ii) The case $I(X; Y) = 0$ implies that $H(X) = H(X|Y)$, which is possible if and only if X and Y are statistically independent.
- (iii) Since $H(X|Y)$ is the average amount of uncertainty (information) in X after we observe Y and $H(X)$ is the average amount of uncertainty (self information) of X , $I(X; Y)$ is the average amount of uncertainty (mutual information) about X having observed Y .
- (iv) Since $H(X) \geq H(X|Y)$, the observation of Y does not increase the entropy (uncertainty). It can only decrease the entropy. That is, observing Y cannot reduce the information about X , it can only add to the information. Thus, conditioning does not increase entropy.

Definition 1.7 The **Joint Entropy** of a pair of discrete random variables (X, Y) with a joint distribution $p(x, y)$ is defined as

$$H(X, Y) = -\sum_{i=1}^n \sum_{j=1}^m p(x_i, y_j) \log p(x_i, y_j) \quad (1.23)$$

By using the mathematical definitions of $H(X)$, $H(X, Y)$ and $H(X|Y)$ we obtain the following **chain rule**:

$$H(X, Y) = H(X) + H(Y|X) = H(Y) + H(X|Y). \quad (1.24)$$

From (1.22) and (1.24) we obtain

$$I(X; Y) = H(X) + H(Y) - H(X, Y) \quad (1.25)$$

Finally, we note that

$$I(X; X) = H(X) - H(X|X) = H(X) \quad (1.26)$$

This explains why, in Definition 1.5, the average self-information is also called the entropy. The relationship between entropy and the mutual information can be expressed neatly using a Venn diagram, as depicted in Fig. 1.5.

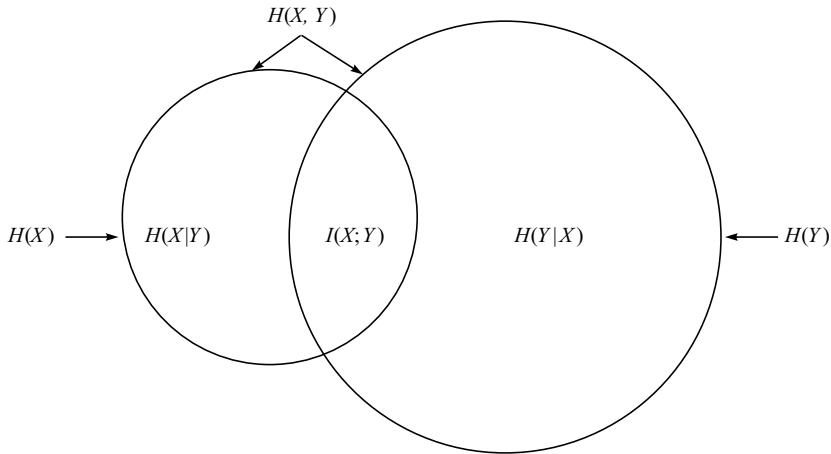


Fig. 1.5 The relationship between entropy and the mutual information.

Example 1.8 Consider the BSC discussed in Example 1.3. Let the input symbols be ‘0’ with probability q and ‘1’ with probability $1 - q$ as shown in Fig. 1.6. Please note that we are dealing with two sets of probabilities:

- DID YOU KNOW ?**
- (i) The source symbol probabilities (input probabilities to the channel): $\{q, 1 - q\}$
 - (ii) The BSC crossover probabilities (channel transition probabilities): $\{p, 1 - p\}$

These two sets of probabilities are independent.

The entropy of this binary source is

$$H(X) = -\sum_{i=0}^1 P(x_i) \log P(x_i) = -q \log_2(q) - (1 - q) \log_2(1 - q).$$

The conditional entropy is given by

$$H(X|Y) = \sum_{i=1}^n \sum_{j=1}^m P(x_i, y_j) \log \frac{1}{P(x_i|y_j)} \quad (1.27)$$

In order to calculate the values of $H(X|Y)$, we can make use of the following equalities:

$$P(x_i, y_j) = P(x_i|y_j)P(y_j) = P(y_j|x_i)P(x_i) \quad (1.28)$$

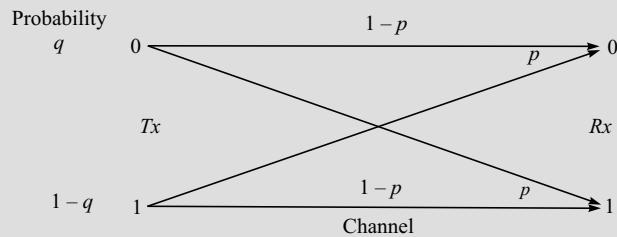


Fig. 1.6 A binary symmetric channel (BSC) with input symbols probabilities equal to q and $1 - q$.

The plot of $H(X|Y)$ versus q is given in Fig. 1.7 with p as the parameter.

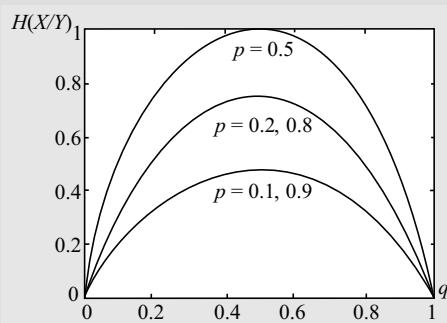


Fig. 1.7 The plot of the conditional entropy $H(X|Y)$ versus q .

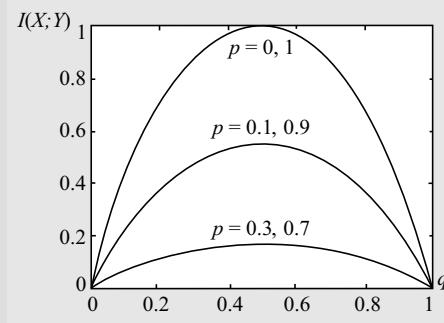


Fig. 1.8 The plot of the average mutual information $I(X; Y)$ versus q .

The average mutual information $I(X; Y)$ is given in Fig. 1.8. It can be seen from the plot that as we increase the parameter p from 0 to 0.5, $I(X; Y)$ decreases. Physically it implies that, as we make the channel less reliable (increase the value of $p \leq 0.5$), the mutual information between the random variable X (at the transmitter end) and the random variable Y (receiver end) decreases.

1.4 Information Measures for Continuous Random Variables

LO 1

The definitions of mutual information for discrete random variables can be directly extended to continuous random variables. Let X and Y be random variables with joint probability density function (pdf) $p(x, y)$ and marginal pdfs $p(x)$ and $p(y)$. The average mutual information between X and Y is defined as follows.

Definition 1.8 The Average Mutual Information between two continuous random variables X and Y is defined as

$$I(X; Y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p(x)p(y|x) \log \frac{p(y|x)p(x)}{p(x)p(y)} dx dy \quad (1.29)$$

It should be pointed out that the definition of average mutual information can be carried over from discrete random variables to continuous random variables, but the concept and physical interpretation cannot. The reason is that the information content in a continuous random variable



is actually infinite, and we require infinite number of bits to represent a continuous random variable precisely. The self information, and hence the entropy, is infinite. To get around the problem we define a quantity called the differential entropy.

Definition 1.9 The **Differential Entropy** of a continuous random variable X is defined as

$$h(X) = - \int_{-\infty}^{\infty} p(x) \log p(x) dx \quad (1.30)$$

Again, it should be understood that there is no physical meaning attached to the above quantity. We carry on with extending our definitions further.

Definition 1.10 The **Average Conditional Entropy** of a continuous random variable X given Y is defined as

$$h(X|Y) = - \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p(x,y) \log p(x|y) dx dy \quad (1.31)$$

The average mutual information can be expressed as

$$I(X; Y) = h(X) - h(X|Y) = h(Y) - h(Y|X) \quad (1.32)$$

Following is the list of some properties of differential entropy:



1. $h(X_1, X_2, \dots, X_n) = \sum_{i=1}^n h(X_i | X_1, X_2, \dots, X_{i-1})$. This is the chain rule for differential entropy.
2. $h(X + c) = h(X)$, i.e., translation does not alter the differential entropy.
3. $h(aX) = h(X) + \log|a|$.
4. If X and Y are independent, then $h(X + Y) \geq h(X)$. This is because $h(X + Y) \geq h(X + Y | Y) = h(X | Y) = h(X)$.

LO 1

An interesting question to ask is how similar (or different) are two probability distributions? Relative entropy is used as a measure of distance between two distributions.



Interpretation

Definition 1.11 The **Relative Entropy** or **Kullback Leibler (KL) Distance** between two probability mass functions $p(x)$ and $q(x)$ is defined as

$$D(p\|q) = \sum_{x \in X} p(x) \log \left(\frac{p(x)}{q(x)} \right) \quad (1.33)$$

It can be interpreted as the expected value of $\log \left(\frac{p(x)}{q(x)} \right)$.

Example 1.9 Consider a Gaussian distribution $p(x)$ with mean and variance given by (μ_1, σ_1^2) , and another Gaussian distribution $q(x)$ with mean and variance given by (μ_2, σ_2^2) . Using (1.33), we can find the KL distance between two Gaussian distributions as

$$D(p\|q) = \frac{1}{2} \left[\frac{\sigma_1^2}{\sigma_2^2} + \left(\frac{\mu_2 - \mu_1}{\sigma_2} \right)^2 - 1 - \log_2 \left(\frac{\sigma_1^2}{\sigma_2^2} \right) \right] \quad (1.34)$$

The distance becomes zero when the two distributions are identical, i.e., $\mu_1 = \mu_2$ and $\sigma_1^2 = \sigma_2^2$. It is interesting to note that when $\mu_1 \neq \mu_2$, the distance is minimum for $\sigma_1^2 = \sigma_2^2$. This minimum distance is given by

$$D_{\min}(p\|q) = \frac{1}{2} \left(\frac{\mu_2 - \mu_1}{\sigma_2} \right)^2$$

Also, the KL distance is infinite if either $\sigma_1^2 \rightarrow 0$ or $\sigma_2^2 \rightarrow 0$, that is, if either of the distributions tends to the Dirac delta.

The average mutual information can be seen as the relative entropy between the joint distribution, $p(x, y)$, and the product distribution, $p(x)p(y)$, i.e.,

$$I(X; Y) = D(p(x, y) \| p(x)p(y)) \quad (1.35)$$

We note that, in general, $D(p\|q) \neq D(q\|p)$. Thus, even though the relative entropy is a distance measure, it does not follow the symmetry property of distances. To overcome this, another measure, called the Jensen Shannon distance, is sometimes used to define the similarity between two distributions.

Definition 1.12 The **Jensen Shannon Distance** between two probability mass functions $p(x)$ and $q(x)$ is defined as

$$JSD(p\|q) = \frac{1}{2} D(p\|m) + \frac{1}{2} D(q\|m) \quad (1.36)$$

where $m = \frac{1}{2}(p + q)$.

DID YOU KNOW ? If the base of the logarithm is 2, then, $0 \leq JSD(p\|q) \leq 1$. Note that Jensen Shannon distance is sometimes referred to as **Jensen Shannon divergence** or **Information Radius** in literature.

Definition 1.13 A function f is said to be **convex** if

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2) \quad (1.37)$$

for all $0 \leq \lambda \leq 1$ and all x_1 and x_2 contained in the convex domain of f . On the other hand, for a function to be concave, it must satisfy

$$f(\lambda x_1 + (1-\lambda)x_2) \geq \lambda f(x_1) + (1-\lambda)f(x_2) \quad (1.38)$$

Replacing ' \leq ' by ' $<$ ' in (1.37) and ' \geq ' by ' $>$ ' in (1.38) would render the function **strictly convex** and **strictly concave** respectively. For any convex function f , we have the **Jensen's inequality** given by $E[f(X)] \geq f(E[X])$ where $E[\cdot]$ is the expectation operator.

One can show that the relative entropy $D(p\|q)$ is convex in the pair (p, q) , while the entropy function $H(p)$ is a concave function of p . The concavity of $H(p)$ is also evident from observing the binary entropy function plotted in Fig. 1.4. Thus, we can write

$$H(\lambda p_1 + (1-\lambda)p_2) \geq \lambda H(p_1) + (1-\lambda)H(p_2) \quad (1.39)$$

Interpreting (1.39) as the addition (mixing) of two systems with different entropies, the resultant entropy is always larger. We also know from thermodynamics that mixing of two ideal gases of equal entropy results in a gas with higher entropy due to the irreversible processes of expansion of the two gases! Again, consider

X and Y to be two independent real-valued random variables. Let Z be a third random variable defined by $Z = X + Y$ (that is, we add results of experiments X and Y together). One can show that $H(Z) \geq H(X)$ and $H(Z) \geq H(Y)$. Thus addition of independent random variables adds uncertainty, which is intuitive.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 1:

- Consider a DMS with source probabilities $\{0.30, 0.25, 0.20, 0.15, 0.10\}$. Find the source entropy, $H(X)$. S
- A source, X , has an infinitely large set of outputs with probability of occurrence given by $P(x_i) = 2^{-i}$, $i = 1, 2, 3, \dots$. What is the average self information, $H(X)$, of this source? S
- Consider a geometrically distributed random variable X with $P(x_i) = p(1-p)^{i-1}$, $i = 1, 2, 3, \dots$. What is the average self information, $H(X)$, of this source? M
- Consider N identical unfair coins, with the probability of getting a head $P(H) = p$. Suppose these N unfair coins are tossed simultaneously and we are interested in the event of getting at least one head.
 - How much information is conveyed by the failure of this event?
 - What happens when $N \rightarrow \infty$ in (i). Explain what it physically means.
- The following table shows the joint probability distribution of two random variables X and Y with respective values x_i and y_j . Calculate $H(X)$, $H(Y)$, $H(X,Y)$, $H(X|Y)$ and $H(Y|X)$. M

	x_1	x_2	x_3
y_1	1/2	1/8	1/8
y_2	1/8	1/8	0

- Calculate the differential entropy, $H(X)$, of the uniformly distributed random variable X with the pdf: S

$$p(x) = \begin{cases} a^{-1} & (0 \leq x \leq a) \\ 0 & (\text{otherwise}) \end{cases}$$

Plot the differential entropy, $H(X)$, versus the parameter a ($0.1 < a < 10$). Comment on the result.

**If you have successfully solved the above problems,
you have mastered LO 1. Congratulations!**



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/576>



Levels of Difficulty

S **Simple:** Level 1 and Level 2 Category

M **Medium:** Level 3 and Level 4 Category

D **Difficult:** Level 5 and Level 6 Category

1.6 Source Coding Theorem

LO 2



State and prove the source coding theorem and discuss its ramifications.

In this section we wish to explore efficient representation (efficient coding) of symbols generated by a source. The primary motivation is the compression of data due to efficient representation of the symbols. Suppose a discrete memoryless source (DMS) outputs a symbol every t seconds. Each symbol is selected from a finite set of symbols x_i , $i = 1, 2, \dots, L$, occurring with probabilities $P(x_i)$, $i = 1, 2, \dots, L$. The entropy of this DMS in bits per source symbols is

$$H(X) = -\sum_{j=1}^L P(x_j) \log_2 P(x_j) \leq \log_2 L \quad (1.40)$$

The equality holds when the symbols are equally likely. It implies that the average number of bits per source symbol is $H(X)$ and the source rate is $H(X)/t$ bits/s.

Now let us suppose that we wish to represent the 26 letters in the English alphabets using bits. We observe that $2^5 = 32 > 26$. Hence, each of the letters can be uniquely represented using 5 bits. This is an example of a **Fixed Length Code (FLC)**. Each letter has a corresponding 5 bit long **codeword**.

Definition 1.14 A **code** is a set of vectors called **codewords**.

Suppose a discrete memoryless source (DMS) outputs a symbol selected from a finite set of symbols x_i , $i = 1, 2, \dots, L$. The number of binary digits (bits) R required for unique coding when L is a power of 2 is

$$R = \log_2 L \quad (1.41)$$

and, when L is not a power of 2, it is

$$R = \lfloor \log_2 L \rfloor + 1 \quad (1.42)$$



As we saw earlier, to encode the letters of the English alphabet, we need $R = \lfloor \log_2 26 \rfloor + 1 = 5$ bits.

Intuition The fixed length code for the English alphabet suggests that each of the letters in the alphabet is equally important (probable) and hence each one requires 5 bits for representation. However, we know that some of the letters are less common (x, q, z etc.) while some others are more frequently used (s, t, e etc.). It appears that allotting equal number of bits to both the frequently used letters as well as not so commonly used letters is *not* an efficient way of representation (coding). Intuitively, we should represent the more frequently occurring letters by fewer numbers of bits and represent the less frequently occurring letters by larger number of bits. In this manner, if we have to encode a whole page of written text, we might end up using fewer number of bits overall. When the source symbols are not equally probable, a more efficient method is to use a **Variable Length Code (VLC)**.

Example 1.10 Suppose we have only the first eight letters of the English alphabet (A – H) in our vocabulary. The fixed length code for this set of letters would be

Fixed length code

Letter	Codeword	Letter	Codeword
A	000	E	100
B	001	F	101
C	010	G	110
D	011	H	111

A variable length code for the same set of letters can be

Variable length code 1

Letter	Codeword	Letter	Codeword
A	00	E	101
B	010	F	110
C	011	G	1110
D	100	H	1111

Suppose we have to code the series of letters: “A BAD CAB”. The fixed length and the variable length representation of the pseudo sentence would be

Fixed Length Code	000 001 000 011 010 000 001	Total bits = 21
Variable Length Code	00 010 00 100 011 00 010	Total bits = 18

Note that the variable length code uses fewer numbers of bits simply because the letters appearing more frequently in the pseudo sentence are represented with fewer numbers of bits.

We look at yet another variable length code for the first 8 letters of the English alphabet:

Variable length code 2

Letter	Codeword	Letter	Codeword
A	0	E	10
B	1	F	11
C	00	G	000
D	01	H	111

This second variable length code appears to be more efficient in terms of representation of the letters.

Variable Length Code 1	00 010 00 100 011 00 010	Total bits = 18
Variable Length Code 2	0 1001 0001	Total bits = 9



However there is a problem with VLC2. Consider the sequence of bits 0 1001 0001 which is used to represent A BAD CAB in the second variable length coding scheme. We could regroup the bits in a different manner to have [0] [10][0][1] [0][0][01] which translates to A EAB AAD or we can decode the vector as [0] [1][0]/[0][1] [0][0]/[0][1] which stands for A BAAB AAAB ! Obviously there is a problem with the unique decoding of the code. We have no clue where the codeword of one letter (symbol) ends and where the next one begins, since the lengths of the codewords are variable. However, this problem does not exist with the VLC1. It can be seen that no codeword forms the prefix of any other codeword. This is called the **prefix condition**. So, as soon as a sequence of bits corresponding to any one of the possible codewords is detected, we can declare that symbol decoded. Such codes are called **Instantaneous Codes**. There is no decoding delay in these codes. If decoding delays are permitted, we may use **Uniquely Decodable Codes** in which the encoded string could be generated by only one possible input string. However, one may have to wait until the entire string is obtained before decoding even the first symbol. In this example, the VLC2 is not a uniquely decodable code, hence not a code of any practical utility. The VLC1 is uniquely decodable, though less economical in terms of bits per symbol.

Definition 1.15 A **Prefix Code** is one in which no codeword forms the prefix of any other codeword. Such codes are also called **Instantaneous Codes**.

Our objective now is to devise a systematic procedure for constructing uniquely decodable, variable length codes that are efficient in terms of average number of bits per source letter. Let the source output a symbol from a finite set of symbols $x_i, i = 1, 2, \dots, L$, occurring with probabilities $P(x_i), i = 1, 2, \dots, L$. The average number of bits per source letter is defined as

$$\bar{R} = \sum_{i=1}^L n_i P(x_i) \quad (1.43)$$

where n_i is the length of the codeword (in terms of number of bits) for the symbol x_i .

Theorem 1.1 Kraft Inequality: A necessary and sufficient condition for the existence of a binary code with codewords having lengths $n_1 \leq n_2 \leq \dots \leq n_L$ that satisfy the prefix condition is



$$\sum_{k=1}^L 2^{-n_k} \leq 1 \quad (1.44)$$

Proof: First we prove the sufficient condition. Consider a binary tree of order (depth) $n = n_L$. This tree has 2^n terminal nodes as depicted in Fig. 1.9. Let us select any code of order n_1 as the first codeword c_1 . Since no codeword is the prefix of any other codeword (the prefix condition), this choice eliminates 2^{n-n_1} terminal codes. This process continues until the last codeword is assigned at the terminal node $n = n_L$. Consider the node of order $j < L$. The fraction of number of terminal nodes eliminated is

$$\sum_{k=1}^j 2^{-n_k} < \sum_{k=1}^L 2^{-n_k} \leq 1 \quad (1.45)$$

Thus, we have been able to construct a prefix code that is embedded in the full tree of n_L nodes. The nodes that are eliminated are depicted by the dotted arrow lines leading on to them in the figure.

We now prove the necessary condition. We observe that in the code tree of the order $n = n_L$, the number of terminal nodes eliminated from the total number of 2^n terminal nodes is

$$\sum_{k=1}^L 2^{n-n_k} \leq 2^n \quad (1.46)$$

This leads to

$$\sum_{k=1}^L 2^{-n_k} \leq 1 \quad (1.47)$$

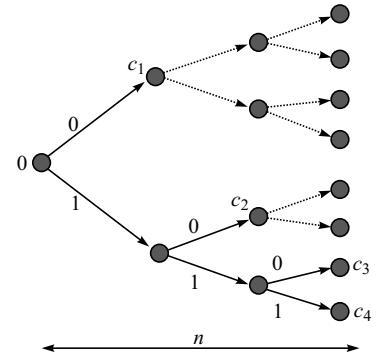


Fig. 1.9 A binary tree of order n_L .

We can easily extend this proof for prefix codes over an alphabet of size M . For the proof we will have to consider an M -ary tree instead of a binary tree. The inequality in this case would become

$$\sum_{k=1}^L M^{-n_k} \leq 1 \quad (1.48)$$

Example 1.11 Consider the construction of a prefix code using a binary tree.

We start from the mother node and proceed toward the terminal nodes of the binary tree (Fig. 1.10). Let the mother node be labeled ‘0’ (could have been labeled ‘1’ as well). There are always two branches emanating from each node (binary tree). Let us label the upper branch ‘0’ and the lower branch ‘1’ (these labels could have also been mutually exchanged). First we follow the upper branch from the mother node. We obtain our first codeword $c_1 = 0$ terminating at node n_{00} . Since we want to construct a prefix code where no codeword is a prefix of any other codeword, we must discard all the daughter nodes generated as a result of the node labeled c_1 . We now proceed on the lower branch from the mother node and reach the node n_{01} . We proceed along the upper branch first and reach node n_{010} . We label this as the codeword $c_2 = 10$ (the labels of the branches that lead up to this node travelling from the mother node). Following the lower branch from the node n_{01} , we ultimately reach the terminal nodes n_{0110} and n_{0111} , which correspond to the codewords $c_3 = 110$ and $c_4 = 111$ respectively. Thus, the binary tree has given us four prefix codewords: {0, 10, 110, 111}. By construction, this is a prefix code. For this code

$$\sum_{k=1}^L 2^{-n_k} = 2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = 0.5 + 0.25 + 0.125 + 0.125 = 1$$

Hence, the Kraft inequality is satisfied.

We now state and prove the noiseless Source Coding theorem, which applies to the codes that satisfy the prefix condition.

Theorem 1.2 Source Coding Theorem: Let X be the ensemble of letters from a DMS with finite entropy $H(X)$ and the output symbols $x_k, k = 1, 2, \dots, L$, occurring with probabilities $P(x_k), k = 1, 2, \dots, L$. It is possible to construct a code that satisfies the prefix condition and has an average length \bar{R} that satisfies the inequality



$$H(X) \leq \bar{R} < H(X) + 1 \quad (1.49)$$

Proof: First consider the lower bound of the inequality. For codewords that have length n_k , $1 \leq k \leq L$, the difference $H(X) - \bar{R}$ can be expressed as

$$H(X) - \bar{R} = \sum_{k=1}^L P(x_k) \log_2 \frac{1}{P(x_k)} - \sum_{k=1}^L P(x_k) n_k = \sum_{k=1}^L P(x_k) \log_2 \frac{2^{-n_k}}{P(x_k)}$$

We now make use of the inequality $\ln x \leq x - 1$ to get

$$\begin{aligned} H(X) - \bar{R} &\leq (\log_2 e) \sum_{k=1}^L P(x_k) \left(\frac{2^{-n_k}}{P(x_k)} - 1 \right) \\ &\leq (\log_2 e) \left(\sum_{k=1}^L 2^{-n_k} - 1 \right) \leq 0 \end{aligned}$$

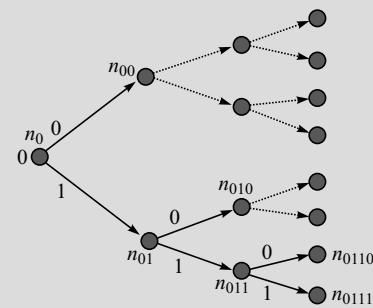


Fig. 1.10 Constructing a binary prefix code using a binary tree.

The last inequality follows from the Kraft inequality. Equality holds if and only if $P(x_k) = 2^{-n_k}$ for $1 \leq k \leq L$. Thus, the lower bound is proved.

We now prove the upper bound. Let us select the codeword lengths n_k such that $2^{-n_k} \leq P(x_k) < 2^{-n_k+1}$. First consider $2^{-n_k} \leq P(x_k)$. Summing both sides over $1 \leq k \leq L$ gives us $\sum_{k=1}^L 2^{-n_k} \leq \sum_{k=1}^L P(x_k) = 1$, which is the Kraft inequality for which there exist a code satisfying the prefix condition. Next consider $P(x_k) < 2^{-n_k+1}$. Take logarithm on both sides to get

$$\log_2 P(x_k) < -n_k + 1$$

or

$$n_k < 1 - \log_2 P(x_k)$$

On multiplying both sides by $P(x_k)$ and summing over $1 \leq k \leq L$ we obtain

$$\sum_{k=1}^L P(x_k) n_k < \sum_{k=1}^L P(x_k) + \left(-\sum_{k=1}^L P(x_k) \log_2 P(x_k) \right)$$

or

$$\bar{R} < H(X) + 1$$

Thus, the upper bound is proved.

The source coding theorem tells us that for any prefix code used to represent the symbols from a source, the *minimum* number of bits required to represent the source symbols *on an average* must be at least equal to the entropy of the source. If we have found a prefix code that satisfies $\bar{R} = H(X)$ for a certain source X , we must abandon further search because we cannot do any better. The theorem also tells us that a source with higher entropy (uncertainty) requires, on an average, more number of bits to represent the source symbols in terms of a prefix code.



Definition 1.16 The **Efficiency** of a prefix code is defined as

$$\eta = \frac{H(X)}{\bar{R}} \quad (1.50)$$

It is clear from the source coding theorem that the efficiency of a prefix code $\eta \leq 1$. Efficient representation of symbols leads to **compression** of data. Source coding is primarily used for compression of data (speech, text, facsimile, image, video etc.).

Example 1.12 Consider a source X which generates four symbols with probabilities $P(x_1) = 0.5$,

$P(x_2) = 0.3$, $P(x_3) = 0.1$ and $P(x_4) = 0.1$. The entropy of this source is

$$H(X) = -\sum_{k=1}^4 P(x_k) \log_2 P(x_k) = 1.685 \text{ bits}$$

Suppose we use the prefix code $\{0, 10, 110, 111\}$ constructed in Example 1.10. Then the average codeword length, \bar{R} is given by

$$\bar{R} = \sum_{k=1}^4 n_k P(x_k) = 1(0.5) + 2(0.3) + 3(0.1) + 3(0.1) = 1.700 \text{ bits}$$

Thus, we have $H(X) \leq \bar{R} \leq H(X) + 1$

The efficiency of this code is $\eta = (1.685/1.700) = 0.9912$. Had the source symbol probabilities been $P(x_k) = 2^{-n_k}$ i.e., $P(x_1) = 2^{-1} = 0.5$, $P(x_2) = 2^{-2} = 0.25$, $P(x_3) = 2^{-3} = 0.125$ and $P(x_4) = 2^{-3} = 0.125$, the average codeword length, $\bar{R} = 1.750$ bits $= H(X)$. In this case, $\eta = 1$.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 2:

1. Determine whether you can construct a binary prefix code with codeword lengths {1, 2, 2, 3}. S
2. Suppose we have to assign binary codewords to N symbols, with each codeword satisfying the prefix condition. Is it possible to have the codeword lengths as {1, 2, 3, ..., N }? S
3. We would like to encode a sequence of symbols that come from an alphabet with $d + 3$ symbols. We want to encode symbols a_1, a_2 , and a_3 using codewords that are three bits long. We want to encode symbols a_4, a_5, \dots, a_{d+3} using codewords that are eight bits long. What is the maximum value of d for which this will be possible, if the code must be uniquely decodable? M
4. Show that the expected length L of an instantaneous D -ary code for a given random variable X can be bounded by $L \geq H_D(X)$, where $H_D(X)$ is the source entropy. When will the equality hold? D

If you have successfully solved the above problems,
you have mastered LO 2. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/577>



1.7 Huffman Coding

We will now study an algorithm for constructing efficient source codes for a DMS with source symbols that are not equally probable. A variable length encoding algorithm was suggested by Huffman in 1952, based on the source symbol probabilities $P(x_i), i = 1, 2, \dots, L$. The algorithm is optimal in the sense that the average number of bits required to represent the source symbols is a minimum provided the prefix condition is met. The steps of the Huffman coding algorithm are as follows:

- (i) Arrange the source symbols in *decreasing* order of their probabilities.
- (ii) Take the bottom two symbols and tie them together as shown in Fig. 1.11. Add the probabilities of the two symbols and write it on the combined node. Label the two branches with a '1' and a '0' as depicted in Fig. 1.11.
- (iii) Treat this sum of probabilities as a new probability associated with a new symbol. Again pick the two smallest probabilities, tie them together to form a new probability. Each time we perform the combination of two symbols we reduce the total number of symbols by one. Whenever we tie together two probabilities (nodes) we label the two branches with a '1' and a '0'.

LO 3

Analyze five source coding techniques: the Huffman encoding, Shannon-Fano-Elias encoding, Arithmetic encoding, the Lempel-Ziv encoding and Run Length encoding.

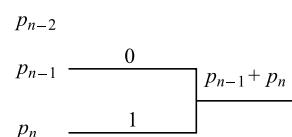


Fig. 1.11 Combining probabilities in Huffman coding.

- (iv) Continue the procedure until only one probability is left (and it should be 1 if your addition is right!). This completes the construction of the **Huffman tree**.
- (v) To find out the prefix codeword for any symbol, follow the branches from the final node back to the symbol. While tracing back the route, read out the labels on the branches. This is the codeword for the symbol.

The algorithm can be easily understood using the following example.

Example 1.13 Consider a DMS with seven possible symbols x_i , $i = 1, 2, \dots, 7$ and the corresponding probabilities $P(x_1) = 0.37$, $P(x_2) = 0.33$, $P(x_3) = 0.16$, $P(x_4) = 0.07$, $P(x_5) = 0.04$, $P(x_6) = 0.02$, and $P(x_7) = 0.01$. We first arrange the probabilities in the decreasing order and then construct the Huffman tree (Fig. 1.12).

Symbol	Probability	Self Information	Codeword
x_1	0.37	1.4344	0
x_2	0.33	1.5995	10
x_3	0.16	2.6439	110
x_4	0.07	3.8365	1110
x_5	0.04	4.6439	11110
x_6	0.02	5.6439	111110
x_7	0.01	6.6439	111111

To find the codeword for any particular symbol, we just trace back the route from the final node to the symbol. For the sake of illustration we show the route for the symbol x_4 with probability 0.07 with the dotted line. We read out the labels of the branches on the way to obtain the codeword as 1110.

The entropy of the source is found out to be

$$H(X) = -\sum_{k=1}^7 P(x_k) \log_2 P(x_k) = 2.1152 \text{ bits}$$

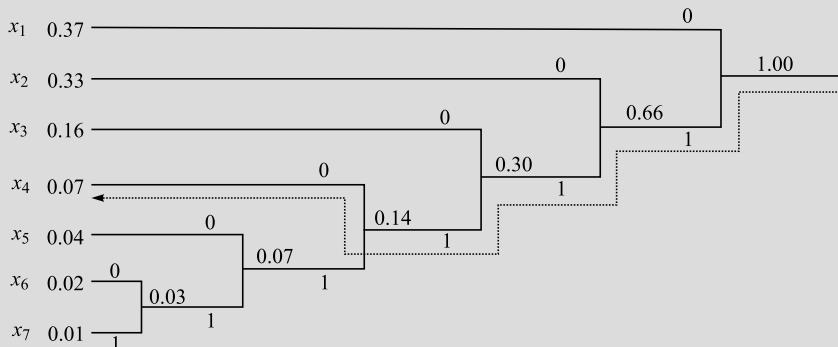


Fig. 1.12 Huffman coding for Example 1.13.

and the average number of binary digits per symbol is calculated to be

$$\begin{aligned}\bar{R} &= \sum_{k=1}^7 n_k P(x_k) \\ &= 1(0.37) + 2(0.33) + 3(0.16) + 4(0.07) + 5(0.04) + 6(0.02) + 6(0.01) = 2.1700 \text{ bits}\end{aligned}$$

The efficiency of this code is $\eta = (2.1152/2.1700) = 0.9747$.

Example 1.14 This example shows that Huffman coding is not unique. Consider a DMS with seven possible symbols x_i , $i = 1, 2, \dots, 7$ and the corresponding probabilities $P(x_1) = 0.46$, $P(x_2) = 0.30$, $P(x_3) = 0.12$, $P(x_4) = 0.06$, $P(x_5) = 0.03$, $P(x_6) = 0.02$, and $P(x_7) = 0.01$.

Symbol	Probability	Self Information	Codeword	Codeword Length
x_1	0.46	1.1203	1	1
x_2	0.30	1.7370	00	2
x_3	0.12	3.0589	010	3
x_4	0.06	4.0589	0110	4
x_5	0.03	5.0589	01110	5
x_6	0.02	5.6439	011110	6
x_7	0.01	6.6439	011111	6

The entropy of the source is found out to be

$$H(X) = -\sum_{k=1}^7 P(x_k) \log_2 P(x_k) = 1.9781 \text{ bits}$$

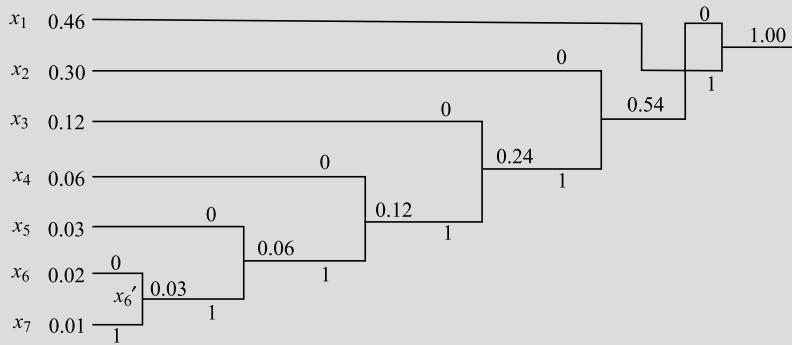


Fig. 1.13 Huffman coding for Example 1.14.

and the average number of binary digits per symbol is calculated to be

$$\begin{aligned} \bar{R} &= \sum_{k=1}^7 n_k P(x_k) \\ &= 1(0.46) + 2(0.30) + 3(0.12) + 4(0.06) + 5(0.03) + 6(0.02) + 6(0.01) = 1.9900 \text{ bits} \end{aligned}$$

The efficiency of this code is $\eta = (1.9781/1.9900) = 0.9940$.

DID YOU KNOW ? We shall now see that Huffman coding is not unique. Consider the combination of the two smallest probabilities (symbols x_6 and x_7). Their sum is equal to 0.03, which is equal to the next higher probability corresponding to the symbol x_5 . So, for the second step, we may choose to put this combined probability (belonging to, say, symbol x_6') higher than, or lower than, the

symbol x_5 . Suppose we put the combined probability at a lower level. We proceed further, to again find the combination of x_6' and x_5 yield the probability 0.06, which is equal to that of symbol x_4 . We again have a choice whether to put the combined probability higher than, or lower than, the symbol x_4 . Each time we make a choice (or flip a fair coin) we end up changing the final codeword for the symbols. In Fig. 1.14, each time we have to make a choice between two probabilities that are equal, we put the probability of the combined symbols at a higher level.

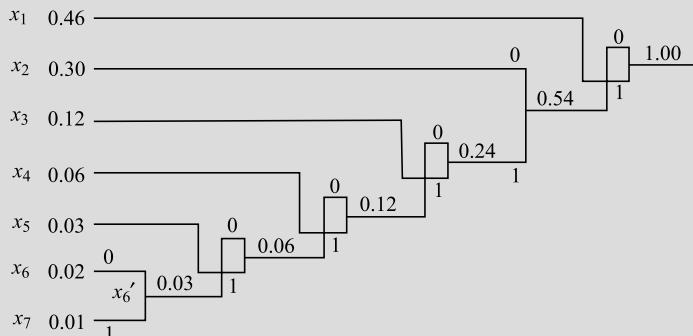


Fig. 1.14 Alternate way of Huffman coding in Example 1.14 which leads to a different code.

Symbol	Probability	Self Information	Codeword	Codeword Length
x_1	0.46	1.1203	1	1
x_2	0.30	1.7370	00	2
x_3	0.12	3.0589	011	3
x_4	0.06	4.0589	0101	4
x_5	0.03	5.0589	01001	5
x_6	0.02	5.6439	010000	6
x_7	0.01	6.6439	010001	6

The entropy of the source is found out to be

$$H(X) = -\sum_{k=1}^7 P(x_k) \log_2 P(x_k) = 1.9781 \text{ bits}$$

and the average number of binary digits per symbol is calculated to be

$$\begin{aligned} \bar{R} &= \sum_{k=1}^7 n_k P(x_k) \\ &= 1(0.46) + 2(0.30) + 3(0.12) + 4(0.06) + 5(0.03) + 6(0.02) + 6(0.01) = 1.9900 \text{ bits} \end{aligned}$$

The efficiency of this code is $\eta = (1.9781 / 1.9900) = 0.9940$. Thus both the codes are equally efficient. This also tells us that Huffman codes are not unique for a given set of probabilities.

Suppose we change the probabilities slightly and the new probabilities are given by $P(x_1) = 0.42$, $P(x_2) = 0.30$, $P(x_3) = 0.15$, $P(x_4) = 0.07$, $P(x_5) = 0.03$, $P(x_6) = 0.02$, and $P(x_7) = 0.01$.

The entropy of this source is $H(X) = -\sum_{k=1}^7 P(x_k) \log_2 P(x_k) = 2.0569$ bits. The Huffman code is given by

Symbol	Probability	Self Information	Codeword	Codeword Length
x_1	0.42	1.2515	1	1
x_2	0.30	1.7370	00	2
x_3	0.15	2.7370	010	3
x_4	0.07	3.8365	0110	4
x_5	0.03	5.0589	01110	5
x_6	0.02	5.6439	011110	6
x_7	0.01	6.6439	011111	6

We observe that in spite of the change in the probability values, the Huffman code is the same as generated by Fig. 1.13. The average codeword length $\bar{R} = \sum_{k=1}^7 n_k P(x_k) = 2.0800$ bits. The efficiency of this code is $\eta = (2.0569/2.0800) = 0.9889$. Intuitively, we can say that the codeword lengths, which *must* be integers, could not change to reflect the change in probabilities (and hence the self information) of the input signals.

Let us now carry out a similar exercise for another set of symbol probabilities as shown in the given table.

Symbol	Probability	Self Information	Codeword	Codeword Length
x_1	$1/2$	1	1	1
x_2	$1/2^2$	2	00	2
x_3	$1/2^3$	3	010	3
x_4	$1/2^4$	4	0110	4
x_5	$1/2^5$	5	01110	5
x_6	$1/2^6$	6	011110	6
x_7	$1/2^6$	6	011111	6



A probability distribution is called **D-adic** with respect to D if each of the probabilities is equal to D^{-n} for some integer n . By definition, the distribution in the above table is D -adic.

Interpretation

The entropy of this source is $H(X) = -\sum_{k=1}^7 P(x_k) \log_2 P(x_k) = 1.9688$ bits. The average number of binary digits per symbol is $\bar{R} = \sum_{k=1}^7 n_k P(x_k) = 1.9688$ bits. Thus the efficiency of the code, $\eta = 1$. The

Huffman coding scheme has been able to match the codeword lengths *exactly* to the probabilities (and thus the self information) of the symbols. For example, if the self information of x_5 is 5 bits, the Huffman code has actually allocated a codeword of length 5 to this symbol. It is clear that to achieve optimality ($\eta = 1$) the self information of the symbols must be integers, which in turn, implies that the probabilities must be negative powers of 2. This is not always true in the real-world. A more efficient way to match the codeword lengths to the symbol probabilities is done using Arithmetic code, which we will study later in this chapter.

In the above examples, encoding is done symbol by symbol. A more efficient procedure is to encode blocks of B symbols at a time. In this case the bounds of the source coding theorem becomes

$$BH(X) \leq \bar{R}_B < BH(X) + 1 \quad (1.51)$$

since the entropy of a B -symbol block is simply $BH(X)$, and \bar{R}_B is the average number of bits per B -symbol block. We can rewrite the bound as

$$H(X) \leq \frac{\bar{R}_B}{B} < H(X) + \frac{1}{B} \quad (1.52)$$

where $\frac{\bar{R}_B}{B} \equiv \bar{R}$ is the average number of bits per source symbol.



Thus, \bar{R} can be made arbitrarily close to $H(X)$ by selecting a large enough block B .

Example 1.15

Consider the source symbols and their respective probabilities listed in the given table.

Symbol	Probability	Self Information	Codeword
x_1	0.40	1.3219	1
x_2	0.35	1.5146	00
x_3	0.25	2.0000	01

For this code, the entropy of the source is

$$H(X) = -\sum_{k=1}^3 P(x_k) \log_2 P(x_k) = 1.5589 \text{ bits}$$

The average number of binary digits per symbol is

$$\bar{R} = \sum_{k=1}^3 n_k P(x_k) = 1(0.40) + 2(0.35) + 2(0.25) = 1.60 \text{ bits}$$

and the efficiency of this code is $\eta = (1.5589/1.6000) = 0.9743$.

We now group together the symbols, two at a time, and again apply the Huffman encoding algorithm. The probabilities of the symbol pairs, in decreasing order, are listed in the table.

Symbol Pairs	Probability	Self Information	Codeword
x_1x_1	0.1600	2.6439	10
x_1x_2	0.1400	2.8365	001
x_2x_1	0.1400	2.8365	010
x_2x_2	0.1225	3.0291	011
x_1x_3	0.1000	3.3219	111
x_3x_1	0.1000	3.3219	0000
x_2x_3	0.0875	3.5146	0001
x_3x_2	0.0875	3.5146	1100
x_3x_3	0.0625	4.0000	1101

For this code, the entropy is

$$2H(X) = -\sum_{k=1}^9 P(x_k) \log_2 P(x_k) = 3.1177 \text{ bits}$$

$$\Rightarrow H(X) = 1.5589 \text{ bits}$$

Note that the source entropy has not changed! The average number of binary digits per block (symbol pair) is

$$\begin{aligned} \bar{R}_B &= \sum_{k=1}^9 n_k P(x_k) \\ &= 2(0.1600) + 3(0.1400) + 3(0.1400) + 3(0.1225) + 3(0.1000) + \\ &\quad 4(0.1000) + 4(0.0875) + 4(0.0875) + 4(0.0625) \\ &= 3.1775 \text{ bits per symbol pair} \end{aligned}$$

$$\Rightarrow \bar{R} = 3.1775/2 = 1.5888 \text{ bits per symbol}$$

and the efficiency of this code is $\eta = (1.5589/1.5888) = 0.9812$. Thus we see that grouping of two letters to make a symbol has improved the coding efficiency.

Example 1.16 Consider the source symbols and their respective probabilities listed in the table.

Symbol	Probability	Self Information	Codeword
x_1	0.50	1.0000	1
x_2	0.30	1.7370	00
x_3	0.20	2.3219	01

For this code, the entropy of the source is

$$H(X) = -\sum_{k=1}^3 P(x_k) \log_2 P(x_k) = 1.4855 \text{ bits}$$

The average number of binary digits per symbol is

$$\bar{R} = \sum_{k=1}^3 n_k P(x_k) = 1(0.50) + 2(0.30) + 2(0.20) = 1.50 \text{ bits}$$

and the efficiency of this code is $\eta = (1.4855 / 1.5000) = 0.9903$.

We now group together the symbols, two at a time, and again apply the Huffman encoding algorithm. The probabilities of the symbol pairs, in decreasing order, are listed in the table.

Symbol Pairs	Probability	Self Information	Codeword
x_1x_1	0.25	2.0000	00
x_1x_2	0.15	2.7370	010
x_2x_1	0.15	2.7370	011
x_1x_3	0.10	3.3219	100
x_3x_1	0.10	3.3219	110
x_2x_2	0.09	3.4739	1010
x_2x_3	0.06	4.0589	1011
x_3x_2	0.06	4.0589	1110
x_3x_3	0.04	4.6439	1111

For this code, the entropy is

$$2H(X) = -\sum_{k=1}^9 P(x_k) \log_2 P(x_k) = 2.9710 \text{ bits}$$

$$\Rightarrow H(X) = 1.4855 \text{ bits}$$

The average number of binary digits per block (symbol pair) is

$$\bar{R}_B = \sum_{k=1}^9 n_k P(x_k)$$

$$= 2(0.25) + 3(0.15) + 3(0.15) + 3(0.10) + 3(0.10) + 4(0.09) + 4(0.06) +$$

$$4(0.06) + 4(0.04) = 3.00 \text{ bits per symbol pair}$$

$$\Rightarrow \bar{R} = 3.00/2 = 1.5000 \text{ bits per symbol}$$

and the efficiency of this code is $\eta_2 = (1.4855 / 1.5000) = 0.9903$. In this case, grouping together two letters at a time has not increased the efficiency of the code! However, if we group 3 letters at a time (triplets) and then apply Huffman coding, we obtain the code efficiency as $\eta_3 = 0.9932$. Upon grouping four letters at a time we see a further improvement ($\eta_4 = 0.9946$).

1.8 Shannon-Fano-Elias Coding

LO 3

Codes that use the codeword lengths of $l(x) = \left\lceil \log \frac{1}{P(x)} \right\rceil$ are called **Shannon Codes**. Shannon codeword

lengths satisfy the Kraft inequality and can therefore be used to construct a uniquely decodable code. In this section we will study another simple method for constructing uniquely decodable codes based on Shannon-Fano-Elias encoding technique. It uses the **Cumulative Distribution Function** to allocate the codewords. The cumulative distribution function is defined as

$$F(x) = \sum_{z \leq x} P(z) \quad (1.53)$$

where $P(z)$ are the probability of occurrence of the z . The cumulative distribution function consists of steps of size $P(x)$, as shown in Fig. 1.15. Let us define a modified cumulative distribution function as

$$\bar{F}(x) = \sum_{z < x} P(z) + \frac{1}{2} P(x) \quad (1.54)$$

where $\bar{F}(x)$ represents the sum of the probabilities of all symbols less than x plus half the probability of the symbols x . The value of the function $\bar{F}(x)$ is the midpoint of the step corresponding to x of the cumulative distribution function. Since probabilities are positive, $F(x) \neq F(y)$ if $x \neq y$. Thus it is possible to determine x given $\bar{F}(x)$ merely by looking at the graph of the cumulative distribution function. Therefore, the value of $\bar{F}(x)$ can be used to code x .

In general, $\bar{F}(x)$ is a real number. This means we require an infinite number of bits to represent $\bar{F}(x)$, which would lead to an inefficient code. Suppose we round off $\bar{F}(x)$ and use only the first $l(x)$ bits, denoted by $\lfloor \bar{F}(x) \rfloor_{l(x)}$. By the definition of rounding off we have

$$\bar{F}(x) - \lfloor \bar{F}(x) \rfloor_{l(x)} \leq \frac{1}{2^{l(x)}} \quad (1.55)$$

If $l(x) = \left\lceil \log \frac{1}{P(x)} \right\rceil + 1$, then,

$$\frac{1}{2^{l(x)}} < \frac{P(x)}{2} = \bar{F}(x) - F(x-1). \quad (1.56)$$

This implies that $\lfloor \bar{F}(x) \rfloor_{l(x)}$ lies within the step corresponding to x , and $l(x)$ bits are sufficient to describe x . The interval corresponding to any codeword is of length $2^{-l(x)}$. From (1.55) we see that this interval is less than half the height of the step corresponding to x . Since we use $l(x) = \left\lceil \log \frac{1}{P(x)} \right\rceil + 1$ bits to represent x , the expected length of this code is

$$\bar{R} = \sum_x P(x) l(x) = \sum_x P(x) \left(\left\lceil \log \frac{1}{P(x)} \right\rceil + 1 \right) < H(X) + 2 \quad (1.57)$$

Thus the Shannon-Fano-Elias coding scheme achieves codeword length within two bits of the entropy.

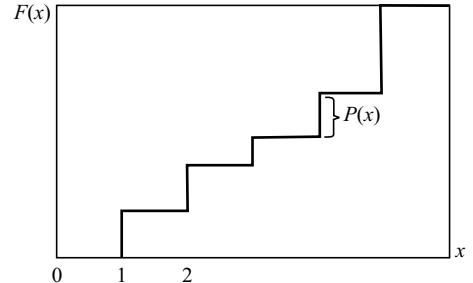


Fig. 1.15 The cumulative distribution function.

Example 1.17 Consider the D -adic distribution given in the following table.

Symbol	Probability	$F(x)$	$\bar{F}(x)$	$\bar{F}(x)$ (binary)	$l(x) = \left\lceil \log \frac{1}{P(x)} \right\rceil + 1$	Codeword
x_1	$1/2$	0.5	0.25	0.01	2	01
x_2	$1/2^2$	0.75	0.625	0.101	3	101
x_3	$1/2^3$	0.875	0.8125	0.1101	4	1101
x_4	$1/2^3$	1	0.9375	0.1111	4	1111

The entropy for this distribution is 1.75 bits. However, the average codeword length for the Shannon-Fano-Elias coding scheme is 2.75 bits. It is easy to observe that if the last bit from all the codewords is deleted, we get the optimal code (Huffman code). It is worthwhile to note that unlike in Huffman coding procedure, here we do not have to arrange the probabilities in descending order first. Let us shuffle the probabilities and redo the exercise.

Symbol	Probability	$F(x)$	$\bar{F}(x)$	$\bar{F}(x)$ (binary)	$l(x) = \left\lceil \log \frac{1}{P(x)} \right\rceil + 1$	Codeword
x_1	$1/2^2$	0.25	0.125	0.001	3	001
x_2	$1/2$	0.75	0.625	0.10	2	10
x_3	$1/2^3$	0.875	0.8125	0.1101	4	1101
x_4	$1/2^3$	1	0.9375	0.1111	4	1111

We observe that the codewords obtained from the Shannon-Fano-Elias coding procedure is not unique. The average codeword length is again 2.75 bits. However, this time we cannot get the optimal code simply by deleting the last bit from every codeword. If we do so, the code no longer remains a prefix code. The basic concept of Shannon-Fano-Elias coding is used in a computationally efficient algorithm for encoding and decoding called Arithmetic Coding.

1.9 Arithmetic Coding

LO 3

As we have seen from Example 1.14, Huffman codes are only optimal if the probabilities of the symbols are negative powers of two. This is because all prefix codes work at the bit level. There are two ways to look at it.



Intuition

- (a) Prefix codes try to match the self information of the symbols using codewords whose lengths are integers. The length matching may ascribe a codeword either longer than the self information or shorter. The exact match is possible if and only if the self information is in integral number of bits.
- (b) If we consider the prefix codes being generated using a binary tree, the decisions between tree branches always take one bit. This is regardless of the fact whether the probabilities for the branches are 0.5/0.5 or 0.9/0.1. In the latter case it would theoretically take only 0.15 bits

($-\log_2(0.9)$) to select the first branch and 3.32 bits ($-\log_2(0.1)$) to select the second branch, making the average code length 0.467 bits ($0.9 \times 0.15 + 0.1 \times 3.32$). The Huffman code still needs one bit for each decision.

Arithmetic coding does not have this restriction. It works by representing the file to be encoded by an interval of real numbers between 0 and 1. Successive symbols in the message reduce this interval in accordance with the probability of that symbol. The more likely symbols reduce the range by less, and thus add fewer bits to the message.

Example 1.18 Let our alphabet consists of only three symbols A , B and C with probabilities of occurrence $P(A) = 0.5$, $P(B) = 0.25$ and $P(C) = 0.25$. We first divide the interval $[0, 1]$ into three intervals proportional to their probabilities, as depicted in step 1 of Fig 1.16. Thus, the variable A corresponds to $[0, 0.5]$, the variable B corresponds to $[0.5, 0.75]$ and the variable C corresponds to $[0.75, 1.0]$. Note that the lengths of these intervals are proportional to their probabilities. Next, suppose the input symbol stream is $B A C A \dots$ We first encode B . This is nothing but choosing the corresponding interval, i.e., $[0.5, 0.75]$. Now, this interval is again subdivided into three intervals, proportional to the probabilities of occurrence. So, for the second step (see Fig. 1.16), the variable A corresponds to $[0.5, 0.625]$, the variable B corresponds to $[0.625, 0.6875]$ and the variable C corresponds to $[0.6875, 1.0]$. Since the next symbol to arrive after B is A , we choose the interval corresponding to A , which is $[0.5, 0.625]$. This is again subdivided to yield the interval $[0.5, 0.5625]$ for A , the interval $[0.5625, 0.59375]$ for B , and the interval $[0.59375, 0.625]$ for C . Now we look at the next symbol to encode, which is C . This corresponds to the interval $[0.59375, 0.625]$. Continuing this process, after encoding A , we are left with the interval $[0.59375, 0.609375]$. The arithmetic code for $B A C A$ is any number that lies within this interval. To complete this example, we can say that the arithmetic code for the sequence $B A C A$ is 0.59375.

Next consider the decoding at the receiver. The receiver needs to know *a-priori*, the probabilities of A , B and C . So, it will also have an identical number line, partitioned into three segments, proportional to the probabilities of A , B and C . Let us say the receiver receives 0.59375. First it checks where this number lies. Clearly, $0.5 < 0.59375 < 0.75$, which is the segment corresponding to B . So the 1st decoded symbol is B . Now we split the segment corresponding to B into three sub-segments proportional to the probabilities of A , B and C (exactly as we did at the encoder side). Again we map the received number 0.59375, and find that it lies in the region of A . The decoding is instantaneous. Mechanically, we proceed to decode the next symbol, and so on. But, how will the receiver know when to stop? Therefore, we need to have a stopping criterion, or a pre-decided protocol to do so.

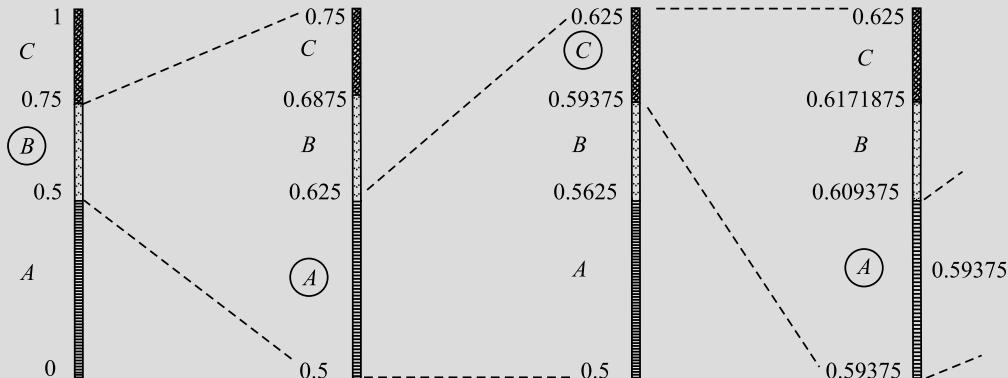


Fig. 1.16 An example to illustrate arithmetic coding of $B A C A$.

1.10 The Lempel-Ziv Algorithm

LO 3

Huffman coding requires symbol probabilities. But most real life scenarios do not provide the symbol probabilities in advance (*i.e.*, the statistics of the source is unknown). In principle, it is possible to observe the output of the source for a long enough time period and estimate the symbol probabilities. However, this is impractical for real-time application. Also, Huffman coding is optimal for a DMS source where the occurrence of one symbol does not alter the probabilities of the subsequent symbols. Huffman coding is not the best choice for a source with memory. For example, consider the problem of compression of written text. We know that many letters occur in pairs or groups, like ‘q-u’, ‘t-h’, ‘i-n-g’ etc. It might be more efficient to use the statistical inter-dependence of the letters in the alphabet along with their individual probabilities of occurrence. Such a scheme was proposed by Lempel and Ziv in 1977. Their source coding algorithm does not need the source statistics. It is a variable-to-fixed length source coding algorithm and belongs to the class of **universal source coding** algorithms.

The logic behind Lempel-Ziv universal coding is as follows. The compression of an arbitrary sequence of bits is possible by coding a series of 0’s and 1’s as some *previous* such string (the prefix string) plus one new bit. Then, the new string formed by adding the new bit to the previously used prefix string becomes a potential prefix string for future strings. These variable length blocks are called **phrases**. The phrases are listed in a dictionary which stores the existing phrases and their locations. In encoding a new phrase, we specify the location of the existing phrase in the dictionary and append the new letter. We can derive a better understanding of how the Lempel-Ziv algorithm works by the following example.

Example 1.19 Suppose we wish to code the string: 101011011010101011. We will begin by **parsing** it into comma-separated phrases that represent strings that can be represented by a previous string as a prefix, plus a bit.

The first bit, a 1, has no predecessors, so, it has a null prefix string and the one extra bit is itself:

1, 01011011010101011

The same goes for the 0 that follows since it can’t be expressed in terms of the only existing prefix:

1, 0, 1011011010101011

So far our dictionary contains the strings ‘1’ and ‘0’. Next we encounter a 1, but it already exists in our dictionary. Hence we proceed further. The following 10 is obviously a combination of the prefix 1 and a 0, so we now have:

1, 0, 10, 11011010101011

Continuing in this way we eventually parse the whole string as follows:

1, 0, 10, 11, 01, 101, 010, 1011

Now, since we found 8 phrases, we will use a three bit code to label the null phrase and the first seven phrases for a total of 8 numbered phrases (with the ninth and last phrase we found being expressed in the others, hence not needing to be numbered). Next, we write the string in terms of the number of the prefix phrase plus the new bit needed to create the new phrase. We will use parentheses and commas to separate these at first, in order to aid our visualization of the process. The eight phrases can be described by:

(000,1)(000,0),(001,0),(001,1),(010,1),(011,1),(101,0),(110,1)

It can be read out as: (codeword at location 0,1), (codeword at location 0,0), (codeword at location 1,0), (codeword at location 1,1), (codeword at location 2,1), (codeword at location 3,1) ...

Thus the coded version of the above string is: 00010000001000110101011110101101. The dictionary for this example is given in Table 1.1. In this case we have not obtained any compression, our coded string is actually longer! However, the larger the initial string, the more saving we get as we move along, because prefixes that are quite large become representable as small numerical indices. In fact, Ziv proved that for long documents, the compression of the file approaches the optimum obtainable as determined by the information content of the document.

Table 1.1 Dictionary for the Lempel-Ziv algorithm.

Dictionary Location	Dictionary Content	Fixed Length Codeword
001	1	0001
010	0	0000
011	10	0010
100	11	0011
101	01	0101
110	101	0111
111	010	1010
-	1011	1101

The next question is what should be the length of the table. In practical application, regardless of the length of the table, it will eventually overflow. This problem can be solved by pre-deciding a large enough size of the dictionary. The encoder and decoder can update their dictionaries by periodically substituting the less used phrases from their dictionaries by more frequently used ones.

INDUSTRIAL RELEVANCE



Lempel-Ziv algorithm is widely used in practice. The compress and uncompress utilities of the UNIX operating system use a modified version of this algorithm. The standard algorithms for compressing binary files use codewords of 12 bits and transmit 1 extra bit to indicate a new sequence. Using such a code, the Lempel-Ziv algorithm can compress transmissions of English text by about 55 percent, whereas the Huffman code compresses the transmission by only 43 percent.

DID YOU KNOW The Graphics Interchange Format (GIF) is also based on Lempel-Ziv coding. In GIF, the image is first converted into a long string by traversing the image row-by-row. The Lempel-Ziv coding is then applied to this string. However, the 2-D structure of the image is not used as an advantage. The Portable Network Graphic (PNG) format utilizes the 2-D structure of the image by first subtracting the pixel values of a row from the one above it, and then carrying out Lempel-Ziv coding on the residual.

In the following section we will study another type of source coding scheme, particularly useful for facsimile transmission and image compression.

1.11 Run Length Encoding

LO 3

Run-length Encoding, or RLE is a technique used to reduce the size of a repeating string of characters. This repeating string is called a **run**. Typically RLE encodes a run of symbols into two bytes, a count and a symbol. RLE can compress any type of data regardless of its information content, but the content of data to be compressed affects the compression ratio. RLE cannot achieve high compression ratios compared to other compression methods, but it is easy to implement and is quick to execute. Run-length encoding is supported by most bitmap file formats such as TIFF, JPG, BMP, PCX and fax machines.

Example 1.20 Consider the following bit stream:

$$S = 1111111111111110000000000000000000001111 \quad (1.58)$$

This can be represented as: fifteen 1's, nineteen 0's, four 1's, i.e., (15,1), (19, 0), (4,1). Since the maximum number of repetitions is 19, which can be represented with 5 bits, we can encode the bit stream as (01111,1), (10011,0), (00100,1). The compression ratio in this case is 18:38 = 1:2.11.



RLE is highly suitable for FAX images of typical office documents. These two-color images (black and white) are predominantly white. If we spatially sample these images for conversion into digital data, we find that many horizontal lines are entirely white (long runs of 0's). Furthermore, if a given pixel is black or white, the chances are very good that the next pixel will match. The code for fax machines is actually a combination of a run-length code and a Huffman code. A run-length code maps run lengths into codewords, and the codebook is partitioned into two parts. The first part contains symbols for runs of lengths that are a multiple of 64; the second part is made up of runs from 0 to 63 pixels. Any run length would then be represented as a multiple of 64 plus some remainder. For example, a run of 205 pixels would be sent using the codeword for a run of length 192 (3×64) plus the codeword for a run of length 13. In this way the number of bits needed to represent the run is decreased significantly. In addition, certain runs that are known to have a higher probability of occurrence are encoded into codewords of short length, further reducing the number of bits that need to be transmitted. Using this type of encoding, typical compressions for facsimile transmission range between 4 to 1 and 8 to 1. Coupled to higher modem speeds, these compressions reduce the transmission time of a single page to less than a minute.



INDUSTRIAL RELEVANCE

Run length coding is also used for the compression of images in the PCX format. The PCX format was introduced as part of the PC Paintbrush series of software for image painting and editing, and became one of the first widely accepted DOS imaging standards. The PCX format is actually an umbrella name for several possible image compression methods and a means to identify which has been applied. Today, PCX has been succeeded by more sophisticated image formats, such as GIF, JPEG and PNG.

We will restrict ourselves to that portion of the PCX data stream that actually contains the coded image, and not those parts that store the color palette and image information such as number of lines, pixels per line, file and the coding method.

The basic scheme is as follows. If a string of pixels are identical in color value, encode them as a special **flag byte** which contains the count followed by a byte with the value of the repeated pixel. If the pixel is not repeated, simply encode it as the byte itself. Such simple schemes can often become more complicated in practice. Consider that in the above scheme, if all 256 colors in a palette are used in an image, then, we need all 256 values of a byte to represent those colors. Hence if we are going to use just bytes as our basic code unit, we don't have any possible unused byte values that can be used as a flag/count byte. On the other hand, if we use two bytes for every coded pixel to leave room for the flag/count combinations, we might double the size of pathological images instead of compressing them.

DID YOU KNOW ? The compromise in the PCX format is based on the belief of its designers than many user-created drawings (which was the primary intended output of their software) would not use all 256 colors. So, they optimized their compression scheme for the case of up to 192 colors only. Images with more colors will also probably get good compression, just not quite as good, with this scheme.

Example 1.21 PCX compression encodes single occurrences of color (that is, a pixel that is not part of a run of the same color) 0 through 191 simply as the binary byte representation of exactly that numerical value. Consider the Table 1.2.

Table 1.2 Example of PCX encoding.

<i>Pixel color Value</i>	<i>Hex Code</i>	<i>Binary Code</i>
0	00	00000000
1	01	00000001
2	02	00000010
3	03	00000011
:	:	:
190	BE	10111110
191	BF	10111111

For the color 192 (and all the colors higher than 192), the codeword is equal to one byte in which the two most significant bits (MSBs) are both set to a 1. We will use these codewords to signify a flag and count byte. If the two MSBs are equal to one, we will say that they have flagged a count. The remaining 6 bits in the flag/count byte will be interpreted as a 6 bit binary number for the count (from 0 to 63). This byte is then followed by the byte which represents the color. In fact, if we have a run of pixels of one of the colors with palette code even over 191, we can still code the run easily since the top two bits are not reserved in this second, color code byte of a run coding byte pair.

If a run of pixels exceeds 63 in length, we simply use this code for the first 63 pixels in the run and then code additional runs of that pixel until we exhaust all pixels in the run. The next question is: how do we code those remaining colors in a nearly full palette image when there is no run? We still code these as a run by simply setting the run length to 1. That means, for the case of at most 64 colors which appear as single pixels in the image and not part of runs, we expand the data by a factor of two. Luckily this rarely happens!

In the next section, we will study coding for analog sources. Recall that we ideally need infinite number of bits to accurately represent an analog source. Anything fewer will only be an approximate representation. We can choose to use fewer and fewer bits for representation at the cost of a poorer approximation of the original signal (rule of thumb: there is no free lunch!). Thus, quantization of the amplitudes of the sampled signals results in data compression. We would like to study the distortion introduced when the samples from the information source are quantized.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 3:

1. Consider a DMS with source probabilities $\{0.35, 0.25, 0.20, 0.15, 0.05\}$.
 - (i) Determine the Huffman code for this source. S
 - (ii) Determine the average length \bar{R} of the codewords.

What is the efficiency η of the code?
2. Again consider a DMS with source probabilities $\{0.35, 0.25, 0.20, 0.15, 0.05\}$
 - (i) Determine the ternary Huffman code for this source using symbols 1, 2 and 3. M
 - (ii) Determine the average length \bar{R} of the codewords and compare it with the binary case.
3. Give the Shannon-Fano-Elias code for the probability distribution $\{0.25, 0.25, 0.20, 0.15, 0.15\}$. Compare it with the Huffman code.
4. Determine the Lempel Ziv code for the following bit stream 0100111100101000001010 101100110000. Recover the original sequence from the encoded stream. M

**If you have successfully solved the above problems,
you have mastered LO 3. Congratulations!**



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/578>



1.12 Rate Distortion Function

Although we live in an analog world, most of the communication takes place in the digital form. Since most natural sources (e.g. speech, video etc.) are analog, they are first sampled, quantized and then processed. However, the representation of an arbitrary real number requires an infinite number of bits. Thus, a finite representation of a continuous random variable can never be perfect.

Consider an analog message waveform $x(t)$ which is a sample waveform of a stochastic process $X(t)$. Assuming $X(t)$ is a bandlimited, stationary process, it can be represented by a sequence of uniform samples taken at the Nyquist rate. These samples are quantized in amplitude and encoded as a sequence of binary digits. A simple encoding strategy can be to define L levels and encode every sample using

$$\begin{aligned} R &= \log_2 L \text{ bits if } L \text{ is a power of 2, or} \\ R &= \lfloor \log_2 L \rfloor + 1 \text{ bits if } L \text{ is not a power of 2} \end{aligned} \quad (1.59)$$

If all levels are not equally probable we may use entropy coding for a more efficient representation. In order to represent the analog waveform more accurately, we need more number of levels, which would imply more number of bits per sample. Theoretically we need infinite bits per sample to perfectly represent an analog source. Quantization of amplitude results in data compression at the cost of signal distortion. It's a form of lossy data compression. Distortion implies some measure of difference between the actual source samples $\{x_k\}$ and the corresponding quantized value $\{\tilde{x}_k\}$.

Definition 1.17 The **Squared-Error Distortion** is defined as

$$d(x_k, \tilde{x}_k) = (x_k - \tilde{x}_k)^2 \quad (1.60)$$

In general a distortion measure may be represented as $d(x_k, \tilde{x}_k) = |x_k - \tilde{x}_k|^p$.

The **Hamming Distortion** is defined as

$$d(x_k, \tilde{x}_k) = \begin{cases} 0 & \text{if } x_k = \tilde{x}_k \\ 1 & \text{if } x_k \neq \tilde{x}_k \end{cases} \quad (1.61)$$

Consider a sequence of n samples, X_n , and the corresponding n quantized values, \tilde{X}_n . Let $d(x_k, \tilde{x}_k)$ be the distortion measure per sample (letter). Then the distortion measure between the original sequence and the sequence of quantized values will simply be the average over the n source output samples, i.e.,

$$d(X_n, \tilde{X}_n) = \frac{1}{n} \sum_{k=1}^n d(x_k, \tilde{x}_k) \quad (1.62)$$

We observe that the source is a random process, hence X_n and consequently $d(X_n, \tilde{X}_n)$ are random variables. We now define the distortion as follows.

Definition 1.18 The **Distortion** between a sequence of n samples, X_n , and their corresponding n quantized values, \tilde{X}_n is defined as

$$D = E[d(X_n, \tilde{X}_n)] = \frac{1}{n} \sum_{k=1}^n E[d(x_k, \tilde{x}_k)] = E[d(x_k, \tilde{x}_k)] \quad (1.63)$$

LO 4



Underline the concept of the rate distortion function and the design of the optimum quantizer.

It has been assumed here that the random process is stationary. Next, let a memoryless source have a continuous output X and the quantized output alphabet \tilde{X} . Let the probability density function of this continuous amplitude be $p(x)$ and per letter distortion measure be $d(x, \tilde{x})$, where $x \in X$ and $\tilde{x} \in \tilde{X}$. We next introduce the rate distortion function which gives us the minimum number of bits per sample required to represent the source output symbols given a prespecified allowable distortion.

Definition 1.19 The minimum rate (in bits/source output) required to represent the output X of the memoryless source with a distortion less than or equal to D is called the **Rate Distortion Function** $R(D)$, defined as



$$R(D) = \min_{p(\tilde{x}|x): E[d(X, \tilde{X})] \leq D} I(X; \tilde{X}) \quad (1.64)$$

where $I(X; \tilde{X})$ is the average mutual information between X and \tilde{X} . $R(D)$ is also known as the **Information Rate Distortion Function**.

Some interesting properties of the rate distortion function are as follow:

- (i) $R(D)$ is non-increasing in D
- (ii) $R(D)$ is convex
- (iii) $R(0) \leq H(X)$
- (iv) $R(D) = 0$ for $D \geq D_{\max}$

We will now state (without proof) two theorems related to the rate distortion function.

Theorem 1.3 The minimum information rate necessary to represent the output of a discrete time, continuous amplitude memoryless Gaussian source with variance σ_x^2 , based on a mean square-error distortion measure per symbol, is

$$R_g(D) = \begin{cases} \frac{1}{2} \log_2(\sigma_x^2/D) & 0 \leq D \leq \sigma_x^2 \\ 0 & D > \sigma_x^2 \end{cases} \quad (1.65)$$



Consider the two cases:

- (i) $D \geq \sigma_x^2$: For this case there is no need to transfer any information. For the reconstruction of the samples (with distortion greater than or equal to the variance) one can use statistically independent, zero mean Gaussian noise samples with variance $D - \sigma_x^2$.
- (ii) $D < \sigma_x^2$: For this case the number of bits per output symbol decreases monotonically as D increases. The plot of the rate distortion function is given in Fig. 1.17.

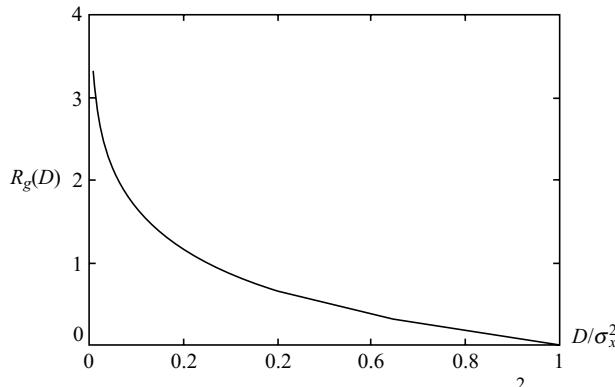


Fig. 1.17 Plot of the $R_g(D)$ versus D/σ_x^2 .

Theorem 1.4 There exists an encoding scheme that maps the source output into codewords such that for any given distortion D , the minimum rate $R(D)$ bits per sample is sufficient to reconstruct the source output with an average distortion that is arbitrarily close to D .

Thus, the distortion function for any source gives the lower bound on the source rate that is possible for a given level of distortion.

Definition 1.20 The **Distortion Rate Function** for a discrete time, memoryless Gaussian source is defined as

$$D_g(R) = 2^{-2R} \sigma_x^2 \quad (1.66)$$

Example 1.22 For a discrete time, memoryless Gaussian source, the distortion (in dB) as a function of its variance can be expressed as

$$10\log_{10} D_g(R) = -6R + 10 \log_{10} \sigma_x^2 \quad (1.67)$$

Thus, the mean square distortion decreases at a rate of 6 dB/bit. In other words, each bit reduces the expected distortion by a factor of 4.

The rate distortion function of a discrete time, memoryless continuous amplitude source with zero mean and finite variance σ_x^2 with respect to the mean square error distortion measure D is upper bounded as

$$R(D) \leq \frac{1}{2} \log_2 (\sigma_x^2 / D) \quad 0 \leq D \leq \sigma_x^2 \quad (1.68)$$

This upper bound can be intuitively understood as follows. We have seen earlier that for a given variance, the zero mean Gaussian random variable exhibits the maximum differential entropy attainable by any random variable. Hence, for a given distortion, the minimum number of bits per sample required is upper bounded by the Gaussian random variable.

The next obvious question is: What would be a good design for a quantizer? Is there a way to construct a quantizer that minimizes the distortion without using too many bits? We shall find the answers to these questions in the next section.

1.13 Optimum Quantizer Design

LO 4

In this section, we look at optimum quantizers design. Consider a continuous amplitude signal whose amplitude is not uniformly distributed, but varies according to a certain probability density function, $p(x)$. We wish to design the optimum scalar quantizer that minimizes some function of the quantization error $q = \tilde{x} - x$, where \tilde{x} is the quantized value of x . The distortion resulting due to the quantization can be expressed as

$$D = \int_{-\infty}^{\infty} f(\tilde{x} - x)p(x)dx \quad (1.69)$$

where $f(\tilde{x} - x)$ is the desired function of the error. An optimum quantizer is one that minimizes D by optimally selecting the output levels and the corresponding input range of each output level. The resulting optimum quantizer is called the *Lloyd-Max quantizer*. For an L -level quantizer the distortion is given by

$$D = \sum_{k=1}^L \int_{x_{k-1}}^{x_k} f(\tilde{x}_k - x)p(x)dx \quad (1.70)$$



The necessary conditions for minimum distortion are obtained by differentiating D with respect to $\{x_k\}$ and $\{\tilde{x}_k\}$. As a result of the differentiation process we end up with the following system of equations

$$\begin{aligned} f(\tilde{x}_k - x_k) &= f(\tilde{x}_{k+1} - x_k), \quad k = 1, 2, \dots, L-1 \\ \int_{x_{k-1}}^{x_k} f'(\tilde{x}_{k+1} - x)p(x)dx, \quad k &= 1, 2, \dots, L \end{aligned} \quad (1.71)$$

For $f(x) = x^2$, i.e., the mean square value of the distortion, the above equations simplify to

$$\begin{aligned} x_k &= \frac{1}{2}(\tilde{x}_k + \tilde{x}_{k+1}), \quad k = 1, 2, \dots, L-1 \\ \int_{x_{k-1}}^{x_k} (\tilde{x}_k - x)p(x)dx &= 0, \quad k = 1, 2, \dots, L \end{aligned} \quad (1.72)$$

The non uniform quantizers are optimized with respect to the distortion. However, each quantized sample is represented by *equal* number of bits (say, R bits/sample). It is possible to have a more efficient variable length coding. The discrete source outputs that result from quantization can be characterized by a set of probabilities p_k . These probabilities can then be used to design efficient variable length codes (source coding). In order to compare the performance of different nonuniform quantizers, we first fix the distortion, D , and then compare the average number of bits required per sample.

Example 1.23 Consider an eight level quantizer for a Gaussian random variable. This problem was first solved by Max in 1960. The random variable has zero mean and variance equal to unity. For a mean square error minimization, the values x_k and \tilde{x}_k are listed in Table 1.3.

Table 1.3 Optimum quantization and Huffman coding.

Level, k	x_k	\tilde{x}_k	$P(x_k)$	Huffman Code
1	-1.748	-2.152	0.040	0010
2	-1.050	-1.344	0.107	011
3	-0.500	-0.756	0.162	010
4	0	-0.245	0.191	10
5	0.500	0.245	0.191	11
6	1.050	0.756	0.162	001
7	1.748	1.344	0.107	0000
8	∞	2.152	0.040	0011

For these values, $D = 0.0345$ which equals -14.62 dB.

The number of bits/sample for this optimum 8-level quantizer is $R = 3$. On performing Huffman coding, the average number of bits per sample required is $R_H = 2.88$ bits/sample. The theoretical limit is $H(X) = 2.82$ bits/sample.

1.14 Entropy Rate of a Stochastic Process

LO 4

First, let us consider a flow of information from an information source generating long sequences of independent symbols. The instantaneous information flow may vary greatly due to the inherent randomness

in the selection of symbols. It would, therefore, be interesting to calculate the average information content of a symbol in a long message. Suppose, the source X uses M symbols s_1, s_2, \dots, s_M with associated probabilities of occurrences p_1, p_2, \dots, p_M . In a long message of length N , the symbol s_i will occur $p_i N$ times. The information content of the i^{th} symbol will be $\log_2\left(\frac{1}{p_i}\right)$ bits. Thus, the $p_i N$ occurrences of s_i will contribute an information content of $p_i N \log_2\left(\frac{1}{p_i}\right)$ bits. Therefore, for all the M symbols, the total contribution will be

$$I_{\text{Tot}} = \sum_{i=1}^M p_i N \log_2\left(\frac{1}{p_i}\right) \text{ bits} \quad (1.73)$$

The average information per symbol for the source X is obtained by

$$H(X) = \frac{I_{\text{Tot}}}{N} = \sum_{i=1}^M p_i \log_2\left(\frac{1}{p_i}\right) \text{ bits} \quad (1.74)$$

This definition of entropy is based on time averaging.

A simple extension of the source coding theorem tells us that $nH(X)$ bits are sufficient, on an average, to describe n independent and identically distributed random variable, each with entropy $H(X)$. But, in the real world, we do encounter random variables that are dependent. What if the random variables form a stationary process?

Definition 1.21 A stochastic process is said to be **stationary** if the joint distribution of any subset of the sequence of random variables is invariant with respect to shifts in time index, i.e.,

$$P(X_1 = x_1, X_2 = x_2, \dots, X_n = x_n) = P(X_{1+m} = x_1, X_{2+m} = x_2, \dots, X_{n+m} = x_n) \quad (1.75)$$

for every shift m and for all $x_1, x_2, \dots, x_n \in X$.

 **Definition 1.22** A discrete stochastic process X_1, X_2, \dots is said to be a **Markov Chain** or a **Markov Process** if, for $n = 1, 2 \dots$

$$P(X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, \dots, X_1 = x_1) = P(X_{n+1} = x_{n+1} | X_n = x_n) \quad (1.76)$$

for all $x_1, x_2, \dots, x_n, x_{n+1} \in X$.

The probability density function of a Markov process can be written as

$$p(x_1, x_2, \dots, x_n) = p(x_1) p(x_2 | x_1) p(x_3 | x_2) \dots p(x_n | x_{n-1}) \quad (1.77)$$

Example 1.24 Suppose random variables X, Y and Z form a Markov chain and we denote it by $X \rightarrow Y \rightarrow Z$. We observe from (1.77) that $p(x, y, z) = p(x)p(y|x)p(z|y) = p(x, y)p(z|y)$. We can write

$$p(x, z | y) = \frac{p(x, y, z)}{p(y)} = \frac{p(x, y)p(z|y)}{p(y)} = p(x|y)p(z|y) \quad (1.78)$$

Thus, $X \rightarrow Y \rightarrow Z$ implies X and Z are conditionally independent given Y . Also note that

$$X \rightarrow Y \rightarrow Z \Leftrightarrow Z \rightarrow Y \rightarrow X \quad (1.79)$$

Interestingly, if $Z = f(Y)$ then $X \rightarrow Y \rightarrow Z$. Using the chain rule we can write

$$I(X; Y, Z) = I(X; Z) + I(X; Y|Z) = I(X; Y) + I(X; Z|Y) \quad (1.80)$$

But, X and Z are conditionally independent given Y and therefore $I(X; Z|Y) = 0$. Therefore

$$I(X; Y) = I(X; Z) + I(X; Y|Z) \quad (1.81)$$

Since, $I(X; Y|Z) \geq 0$, $X \rightarrow Y \rightarrow Z$ implies $I(X; Y) \geq I(X; Z)$. This shows that processing of data (recall $Z = f(Y)$) cannot improve the inferences that can be drawn from it. Manipulating the data (X) can only increase our uncertainty about X . This is also known as the **Data Processing Inequality**.

We now revert back to the entropy rate of a stochastic process. If we have a sequence of n random variables, it is interesting to explore how the entropy of the sequence grows with n . Entropy rate is used to define this rate of growth.

Definition 1.23 The **Entropy Rate** of a stochastic process X is given by

$$H(X) = \lim_{n \rightarrow \infty} \frac{1}{n} H(X_1, X_2, \dots, X_n) \quad (1.82)$$

provided the limit exists.

Example 1.25 Let X_1, X_2, X_3, \dots be independent and identically distributed (i.i.d.) random variables.

In this case the entropy rate would be $H(X) = \lim_{n \rightarrow \infty} \frac{1}{n} H(X_1, X_2, \dots, X_n) = \lim_{n \rightarrow \infty} \frac{nH(X_1)}{n} = H(X_1)$. Thus, the entropy rate of an i.i.d. source is the entropy any of its single symbols.

However, if X_1, X_2, X_3, \dots form a sequence of independent but *not* identically distributed random variables, then $H(X_1, X_2, \dots, X_n) = \sum_{i=1}^n H(X_i)$. Thus, the entropy rate is $\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n H(X_i)$. If $H(X_i)$ are unequal, it is possible to choose a sequence of distributions on X_1, X_2, X_3, \dots such that the limit of $\frac{1}{n} \sum_{i=1}^n H(X_i)$ does not exist.

Definition 1.24 For **Stationary Markov Chain**, the entropy rate is given by

$$H(X) = \lim_{n \rightarrow \infty} H(X_n | X_{n-1}, \dots, X_1) = \lim_{n \rightarrow \infty} H(X_n | X_{n-1}) = H(X_2 | X_1) \quad (1.83)$$

Example 1.26 Consider a two-state Markov chain with a probability transition matrix

$$P = \begin{bmatrix} 1-p_1 & p_1 \\ p_2 & 1-p_2 \end{bmatrix} \quad (1.84)$$

This is depicted in Fig. 1.18.

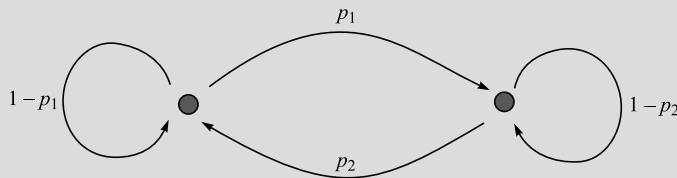


Fig. 1.18 The state transition graph of a two-state Markov chain.

For stationary distribution, the net probability distribution across any cut set in the state transition graph should be zero. Let α and β be the stationary probabilities of the two states. Thus, the stationary distribution is given by

$$\alpha = \frac{p_2}{p_1 + p_2} \quad \text{and} \quad \beta = \frac{p_1}{p_1 + p_2} \quad (1.85)$$

Note that $\alpha + \beta = 1$. The entropy of the state X_n at time n will be

$$H(X_n) = H\left(\frac{p_2}{p_1 + p_2}, \frac{p_1}{p_1 + p_2}\right) = H(\alpha, \beta) \quad (1.86)$$

The entropy rate of this two-state Markov chain is given by

$$H(X) = H(X_2 | X_1) = \frac{p_2}{p_1 + p_2} H(p_1) + \frac{p_1}{p_1 + p_2} H(p_2) \quad (1.87)$$

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 4:

1. Use the Hamming distortion measure to find the rate distortion function $R(D) = \min I(X, \tilde{X})$ for Bernoulli distributed X for a given p . D
2. Let a source $S\{X_k\}$ generate mutually independent symbols X_k , such that $H(X_k) = k$ for $k \geq 1$. Find the entropy rate of this source. S
3. Plot the rate distortion function for a Gaussian source. Verify that $R(D)$ is non-increasing in D , $R(D)$ is convex, $R(0) \leq H(X)$ and $R(D) = 0$ for $D \geq D_{\max}$. M

If you have successfully solved the above problems,
you have mastered LO 4. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/579>



1.15 Introduction to Image Compression

LO 5



Apply the knowledge to study image compression, one of the important application areas of source coding.

Earlier in this chapter we discussed the coding of data sets for compression. By applying these techniques we can store or transmit all of the information content of a string of data with fewer bits than are in the source data. The minimum number of bits that we must use to convey all the information in the source data is determined by the entropy measure of the source. Good compression ratios can be obtained via entropy encoders and universal encoders for sufficiently large source data blocks.

Images can be sampled and quantized sufficiently finely so that a binary data stream can represent the original data to an extent that is satisfactory to the most discerning eye. Since we can represent a picture by something between a thousand and a million bytes of data, we should be able to apply the techniques studied earlier directly to the task of compressing that data for storage and transmission. First, we consider the following points:

DID YOU KNOW

1. High quality images are represented by very large data sets. A photographic quality image may require 40 to 100 million bits for representation. These large file sizes drive the need for extremely high compression ratios to make storage and transmission (particularly of movies) practical.
2. Applications that involve imagery seem to be inherently linked to immediate human consumption, and so need to be fast in execution on computers and in transmission. Television, movies, computer graphical user interfaces, and the World Wide Web are examples of applications in which imagery must be moved from storage or across some kind of distribution network very quickly for immediate human intake.
3. Imagery has the quality of higher redundancy than we can generally expect in arbitrary data. For example, a pair of adjacent horizontal lines in an image are nearly identical (typically), while, two adjacent lines in a book have essentially no commonality.

The first two points indicate that we will almost always want to apply the highest level of compression technology available for the movement and storage of image data. The third factor indicates that compression ratios will usually be quite high. The third factor also says that some special compression techniques may be possible that will take advantage of the structure and properties of image data. The close relationship between neighboring pixels in an image can be exploited to improve the compression ratios. This is very important for the task of coding and decoding image data for real-time applications.

Another interesting point to note is that the human eye is very tolerant to approximation error in an image. Thus, it may be possible to compress the image data in a manner in which the less important information (to the human eye) can be dropped. That is, by trading off some of the quality of the image we might obtain a significantly reduced data size. This technique is called **lossy compression**, as opposed to the **lossless compression** techniques discussed earlier. This sentiment, however, can never be expressed with regards to, say, financial data or even textual data! Lossy compression can only be applied to data such as images and audio for which human beings will tolerate some loss of fidelity.

1.16 The JPEG Standard for Lossless Compression

LO5

INDUSTRIAL RELEVANCE



Let us now consider the lossless compression option of the JPEG image compression standard. The JPEG image compression standard is actually a description of 29 distinct coding systems for compression of images. Why are there so many approaches? It is because the needs of users vary so much with respect to quality versus compression and compression computation time that the committee decided to provide a broad selection from which to choose. We shall briefly discuss here two methods that use entropy coding.

The two lossless JPEG compression options differ only in the form of the entropy code that is applied to the innovations data. The user can choose to use either a **Huffman code** or an **Arithmetic code**.

DID YOU
KNOW ?

We have seen earlier that Arithmetic code, like Huffman code, achieves compression by using the probabilistic nature of the data to render the information with fewer bits than used in the original data stream. Its primary advantage over the Huffman code is that it can come closer to the Shannon entropy limit of compression for data streams that involve a relatively small alphabet. The reason is that Huffman codes work best (highest compression ratios) when the probabilities of the symbols can be expressed as fractions of powers of two. The Arithmetic code construction is not closely tied to these particular values, as is the Huffman code. The computation of coding and decoding Arithmetic codes is more costly than that of Huffman codes. Typically a 5 to 10% reduction in file size is seen with the application of Arithmetic codes over that obtained with Huffman coding.

Some compression can be achieved if we can predict the next pixel using the previous pixels. In this way we just have to transmit the prediction coefficients (or difference in the values) instead of the entire pixel. The predictive process that is used in the lossless JPEG coding schemes to form the innovations data is also variable. However, in this case, the variation is not based upon the user's choice, but rather, for any image on a line-by-line basis. The choice is made according to that prediction method that yields the best prediction overall for the entire line.

There are eight prediction methods available in the JPEG coding standards. One of the eight (which is the no prediction option) is not used for the lossless coding option that we are examining here. The other seven may be divided into the following categories:

1. Predict the next pixel on the line as having the same value as the last one.
2. Predict the next pixel on the line as having the same value as the pixel in this position on the previous line (that is, above it).
3. Predict the next pixel on the line as having a value related to a combination of the previous, above and previous to the above pixel values. One such combination is simply the average of the other three.

DID YOU
KNOW ?

The differential encoding used in the JPEG standard consists of the differences between the actual image pixel values and the predicted values. As a result of the smoothness and general redundancy of most pictures, these differences consist of a series of relatively small positive

and negative numbers that represent the small typical error in the prediction. Hence, the probabilities associated with these values are large for the small innovation values and quite small for large ones. This is exactly the kind of data stream that compresses well with an entropy code.

The typical lossless compression for natural images is 2:1. While this is substantial, it does not in general solve the problem of storing or moving large sequences of images as encountered in high quality video.

1.17 The JPEG Standard for Lossy Compression

LO5

The JPEG standard includes a set of sophisticated lossy compression options which resulted from much experimentation by the creators of JPEG with regard to human acceptance of types of image distortion. The JPEG standard was the result of years of effort by the JPEG which was formed as a joint effort by two large, standing, standards organizations, the CCITT (The European telecommunications standards organization) and the ISO (International Standards Organization).

The JPEG lossy compression algorithm consists of an image simplification stage, which removes the image complexity at some loss of fidelity, followed by a lossless compression step based on predictive filtering and Huffman or Arithmetic coding.

The lossy image simplification step, which we will call the image reduction, is based on the exploitation of an operation known as the **Discrete Cosine Transform** (DCT), defined as follows.

$$Y(k, l) = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} 4y(i, j) \cos\left(\frac{\pi k}{2N}(2i+1)\right) \cos\left(\frac{\pi l}{2M}(2j+1)\right) \quad (1.88)$$

DID YOU KNOW ? where the input image is N pixels by M pixels, $y(i, j)$ is the intensity of the pixel in row i and column j , $Y(k, l)$ is the DCT coefficient in row k and column l of the DCT matrix. All DCT multiplications are real. This lowers the number of required multiplications, as compared to the discrete Fourier transform. For most images, much of the signal energy lies at low frequencies, which appear in the upper left corner of the DCT. The lower right values represent higher frequencies, and are often small (usually small enough to be neglected with little visible distortion). The DCT is, unfortunately, computationally very expensive and its complexity increases as $O(N^2)$. Therefore, images compressed using DCT are first divided into blocks.

In the JPEG image reduction process, the DCT is applied to 8 by 8 pixel blocks of the image. Hence, if the image is 256 by 256 pixels in size, we break it into 32 by 32 square blocks of 8 by 8 pixels and treat each one independently. The 64 pixel values in each block are transformed by the DCT into a new set of 64 values. These new 64 values, known also as the DCT coefficients, form a whole new way of representing an image. The DCT coefficients represent the spatial frequency of the image sub-block. The upper left corner of the DCT matrix has low frequency components and the lower right-corner the high frequency components (see Fig. 1.19). The top left coefficient is called the **DC coefficient**. Its value is proportional to the average value of the 8 by 8 block of pixels. The rest are called the **AC coefficients**.

So far we have not obtained any reduction simply by taking the DCT. However, due to the nature of most natural images, maximum energy (information) lies in low frequency as opposed to high frequency. We can represent the high frequency components coarsely, or drop them altogether, without strongly affecting the quality of the resulting image reconstruction. This leads to a lot of compression (lossy). The JPEG lossy compression algorithm does the following operations:

1. First the lowest weights are trimmed by setting them to zero.
2. The remaining weights are quantized (that is, rounded off to the nearest of some number of discrete code represented values), some more coarsely than others according to observed levels of sensitivity of viewers to these degradations.

DC coefficient	4.32	3.12	3.01	2.41
Low frequency coefficients	2.74	2.11	1.92	1.55
	2.11	1.33	0.32	0.11
	1.62	0.44	0.03	0.02

Fig. 1.19 Typical discrete cosine transform (DCT) values for a 4×4 image block. In JPEG, 8×8 pixel blocks are used.

Now several lossless compression steps are applied to the weight data that results from the above DCT and quantization process, for all the image blocks. We observe that the DC coefficient, which represents the average image intensity, tends to vary slowly from one block of 8×8 pixels to the next. Hence, the prediction of this value from surrounding blocks works well. We just need to send one DC coefficient and the difference between the DC coefficients of successive blocks. These differences can also be source coded.

We next look at the AC coefficients. We first quantize them, which transforms most of the high frequency coefficients to zero. We then use a **zig-zag** coding as shown in Fig. 1.20. The purpose of the zig-zag coding is that we gradually move from the low frequency to high frequency, avoiding abrupt jumps in the values. Zig-zag coding will lead to long runs of 0's, which are ideal for RLE followed by Huffman or Arithmetic coding.

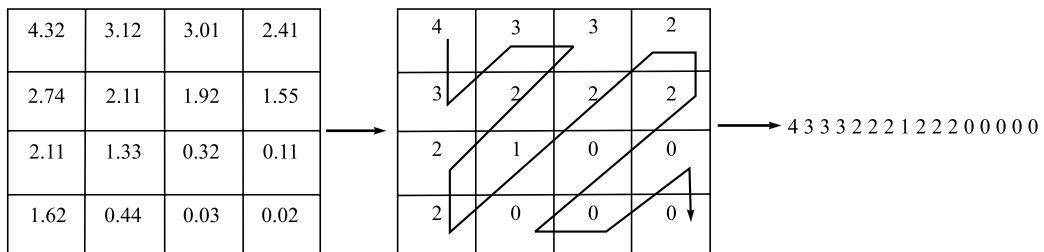


Fig. 1.20 An example of quantization followed by zig-zag coding.

DID YOU KNOW ? The typically quoted performance for JPEG is that photographic quality images of natural scenes can be preserved with compression ratios of up to about 20:1 or 25:1. Usable quality (that is, for non critical purposes) can result for compression ratios in the range of 200:1 up to 230:1.

1.18 Video Compression Standards

LO 5

Consider an uncompressed video with a frame size of 720×576 pixels and a refresh rate of 25 frames per second (fps). Let us represent the luminance value for each pixel using 8 bits per pixel and the chrominance using 8 bits per pixel. Then, the bandwidth requirement for the uncompressed video will be

$$B = 720 \times 576 \times 25 \times 16 = 1.66 \text{ Mb/s} \quad (1.89)$$

For High Definition Television (HDTV) with a frame size of 1920×1080 pixels and a refresh rate of 60 fps, the corresponding bandwidth requirement is

$$B = 1920 \times 1080 \times 60 \times 16 = 1.99 \text{ Mb/s} \quad (1.90)$$

Clearly, uncompressed video produces an enormous amount of data. Fortunately, digital video contains a great deal of inherent redundancy, and it is suitable for compression. There are several lossy compression techniques available for video compression, but there is a trade-off between computational time and quality of the output. Some of the popular video compression standards and the corresponding application areas are given in Table 1.4.

Table 1.4 Popular video compression standards.

H.261	Video conferencing over ISDN
MPEG-1	Video on digital storage media (CD-ROM)
MPEG-2	Digital Television
H.263	Video telephony over PSTN
MPEG-4	Object-based coding, synthetic content, interactivity
H.264	Improved video compression

Here we will briefly discuss H.261 and the MPEG video coding standards.

INDUSTRIAL RELEVANCE



H.261 is an **ITU-T video compression standard** that works in the range of 64 and 384 kbps. It belongs to the H.26x family of video coding standards of the **ITU-T Video Coding Experts Group (VCEG)**. The basic unit of H.261 is the ‘macroblock’, containing four luminance blocks and two chrominance blocks (each 8×8 samples). The supported frame sizes are CIF (352×288 pixels) and QCIF (176×144 pixels). Video data is processed in 4:2:0 Y:Cr:Cb format. The coding algorithm uses a hybrid of motion compensated inter-picture prediction. The motion compensated residual data is coded with an 8×8 DCT followed by quantization. Post quantization, zig-zag coding is carried out, as discussed earlier in the case of JPEG compression. The motion compensation step is improved by use of an (optional) loop filter which is a 2-D spatial filter. This filter operates on each 8×8 block in a macroblock prior to motion compensation. Each macroblock may be coded in ‘intra’ mode (no motion-compensated prediction) or ‘inter’ mode (with motion-compensated prediction). H.261 was developed in the early 1990s at a time when low complexity designs were required due to constraints on hardware and software resources. H.261 has been superseded by H.263, which has higher compression efficiency and greater flexibility. **MPEG** stands for **Moving Picture Coding Experts Group** and encompasses a whole family of international standards for the compression of audio and video digital data.

There are five important steps in MPEG compression:

1. **Reduction of the resolution:** The human eye has a lower sensitivity to color than to intensity (could this be due to the way we have evolved?). Therefore, a conversion from RGB-color-space into YUV-color-space is carried out first. Then color information is represented in fewer numbers of bits than intensity information.
2. **Motion compensation in order to reduce temporal redundancy:** An MPEG video can be interpreted as a sequence of frames. Because of the inherent inertia in the world that we live in, two successive frames of a video sequence often have small differences (except in major scene changes). The MPEG-standard offers a smart way of reducing this temporal redundancy by using three types of frames: I-frames (intra), P-frames (predicted) and B-frames (bidirectional). The I-frames are ‘intra-frames’, which have no reference to other frames and their compression is rather low. The P-frames can be predicted from an earlier I-frame or another P-frame. P-frames cannot be reconstructed without their referencing frame, but they are more compressed than the I-frames, because only the differences are stored. The B-frames are two-directional versions of the P-frame, referring to both directions (one forward frame and one backward frame). The distance between two I-frames can be seen as a measure for the quality of an MPEG-video. Recall that each frame is divided into blocks. To decrease the amount of temporal redundancy in a video, only blocks that change are updated. The MPEG encoder compares the current frame with adjacent parts of the video from the previous I- or P-frame. Only the direction and distance (i.e., the motion vector) is encoded into the inter-frame. This is called motion estimation. The reverse of this process, for reconstruction, is called motion compensation.
3. **Discrete Cosine Transformation (DCT):** Each frame is divided into 8×8 blocks. Each 8×8 block is encoded by first applying a discrete cosine transform given in (1.88). As discussed in the previous section, The upper left corner of the DCT matrix has low frequency components and the lower right-corner contains the high frequency components. After the DCT operation, the ground is set for compression using quantization.
4. **Quantization:** We have seen in the case of JPEG compression that quantization is the primary source of data loss. The human eye is more reactive to low frequencies than to high ones (again, is the biological evolution responsible?). Post quantization, the higher frequency components end up with a zero entry thereby setting up the stage for run-length coding. Again, the quantized AC-terms are stored in a zig-zag-manner with increasing frequency values.
5. **Entropy coding (Run Length Encoding and the Huffman coding algorithm):** The entropy coding has two steps: Run Length Encoding (RLE) and Huffman coding. These have been discussed earlier in the context of JPEG compression.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 5:

1. Consider an image represented by $A = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$. Find its DCT.

M

2. Carry out quantization for the matrix given by $A = \begin{bmatrix} 6.77 & 4.31 & 2.19 & 1.56 & 0.21 \\ 4.02 & 2.44 & 1.83 & 0.31 & 0.01 \\ 2.26 & 2.09 & 0.43 & 0.02 & 0.02 \\ 1.97 & 0.15 & 0.02 & 0.01 & 0 \\ 0.03 & 0.04 & 0.02 & 0.01 & 0 \end{bmatrix}$.

S

Next perform the zig-zag encoding.

3. Perform run length coding on the sequence obtained after zig-zag coding in the previous question.

S

**If you have successfully solved the above problems,
you have mastered LO 5. Congratulations!**



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/580>



1.19 Concluding Remarks

The concept of entropy has its roots in thermodynamics. In 1948, Shannon published his landmark “A Mathematical Theory of Communication”. He begins this pioneering paper on information theory by observing that the fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point. He then proceeds to so thoroughly establish the foundations of information theory that his framework and terminology remain standard. Shannon’s theory was an immediate success with communications engineers and stimulated the technology which led to today’s Information Age. Shannon published many more provocative and influential articles in a variety of disciplines. His master’s thesis, “A Symbolic Analysis of Relay and Switching Circuits”, used Boolean algebra to establish the theoretical underpinnings of digital circuits. This work has broad significance because digital circuits are fundamental to the operation of modern computers and telecommunications systems.

Shannon was renowned for his eclectic interests and capabilities. A favorite story describes him juggling while riding a unicycle down the halls of Bell Labs. He designed and built chess-playing, maze-solving, juggling and mind-reading machines. These activities bear out Shannon's claim that he was more motivated by curiosity than usefulness. In his words "I just wondered how things were put together."

The Kraft inequality for uniquely decodable codes was first proved by McMillan. Therefore, it is sometimes referred to as Kraft-McMillan inequality. The Huffman code was created by American D. A. Huffman in 1952. Modified Huffman coding is today used in the JPEG and MPEG standards. Arithmetic coding, which is an extension of the Shannon-Fano-Elias coding technique, was developed by Rissanen and Pasco and generalized by Rissanen and Langdon.

A very efficient technique for encoding sources without needing to know their probable occurrence was developed in the 1970s by the Israelis Abraham Lempel and Jacob Ziv. The compress and uncompress utilities of the UNIX operating system use a modified version of this algorithm. The GIF format (Graphics Interchange Format), developed by CompuServe, involves simply an application of the Lempel-Ziv-Welch (LZW) universal coding algorithm to the image data. A more recent approach is to use quantum image processing.

And finally, to conclude this chapter we mention that Shannon, the father of Information Theory, passed away on February 24, 2001. Excerpts from the obituary published in the *New York Times*:

Claude Shannon, Mathematician, Dies at 84

By GEORGE JOHNSON

Dr. Claude Elwood Shannon, the American mathematician and computer scientist whose theories laid the groundwork for the electronic communications networks that now lace the earth, died on Saturday in Medford, Mass...

LEARNING OUTCOMES

- The self information of the event $X = x_i$ is given by $I(x_i) = \log\left(\frac{1}{P(x_i)}\right) = -\log P(x_i)$.
- The mutual information $I(x_i; y_j)$ between x_i and y_j is given by $I(x_i; y_j) = \log\left(\frac{P(x_i | y_j)}{P(x_i)}\right)$.
- The conditional self information of the event $X = x_i$ given $Y = y_j$ is defined as

$$I(x_i | y_j) = \log\left(\frac{1}{P(x_i | y_j)}\right) = -\log P(x_i | y_j)$$

- The average mutual information between two random variables X and Y is given by $I(X; Y) = \sum_{i=1}^n \sum_{j=1}^m P(x_i, y_j) I(x_i, y_j) = \sum_{i=1}^n \sum_{j=1}^m P(x_i, y_j) \log \frac{P(x_i, y_j)}{P(x_i)P(y_j)}$. For the case when X and Y are statistically

independent, $I(X; Y) = 0$. The average mutual information $I(X; Y) \geq 0$, with equality if and only if X and Y are statistically independent.

- The average self information of a random variable X is given by

$$H(X) = \sum_{i=1}^n P(x_i) I(x_i) = -\sum_{i=1}^n P(x_i) \log P(x_i). H(X) \text{ is called the entropy.}$$

- The average conditional self information called the conditional entropy is given by

$$H(X|Y) = \sum_{i=1}^n \sum_{j=1}^m P(x_i, y_j) \log \frac{1}{P(x_i|y_j)}.$$

- $I(x_i; y_j) = I(x_i) - I(x_i | y_j)$ and $I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$. Since $I(X; Y) \geq 0$, it implies that $H(X) \geq H(X|Y)$.

- The joint entropy of a pair of discrete random variables (X, Y) with a joint distribution $p(x, y)$ is defined

$$\text{as } H(X, Y) = -\sum_{i=1}^n \sum_{j=1}^m p(x_i, y_j) \log p(x_i, y_j).$$

- The chain rule relating $H(X)$, $H(X, Y)$ and $H(X|Y)$ is $H(X, Y) = H(X) + H(Y|X)$.

- The relationship between entropy and mutual information can be expressed as

$$I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X).$$

- The relationship between mutual information and joint entropy can be expressed as

$$I(X; Y) = H(X) + H(Y) - H(X, Y).$$

- The average mutual information between two continuous random variables X and Y is given by

$$I(X; Y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p(x)p(y|x) \log \frac{p(y|x)p(x)}{p(x)p(y)} dx dy.$$

- The differential entropy of a continuous random variables X is given by $H(X) = - \int_{-\infty}^{\infty} p(x) \log p(x) dx$.

- The average conditional entropy of a continuous random variables X given Y is given by

$$h(X|Y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p(x, y) \log p(x|y) dx dy.$$

- The Relative Entropy or Kullback Leibler (KL) distance between two probability mass functions $p(x)$

$$\text{and } q(x) \text{ is defined as } D(p||q) = \sum_{x \in X} p(x) \log \left(\frac{p(x)}{q(x)} \right).$$

- The Jensen Shannon distance between two probability mass functions $p(x)$ and $q(x)$ is defined as

$$JSD(p||q) = \frac{1}{2} D(p||m) + \frac{1}{2} D(q||m), \text{ where } m = \frac{1}{2}(p+q).$$

- A necessary and sufficient condition for the existence of a binary code with codewords having lengths $n_1 \leq n_2 \leq \dots n_L$ that satisfy the prefix condition is $\sum_{k=1}^L 2^{-n_k} \leq 1$. The efficiency of a prefix code is given by $\eta = \frac{H(X)}{\bar{R}}$.
- Let X be the ensemble of letters from a DMS with finite entropy $H(X)$. The source coding theorem states that it is possible to construct a code that satisfies the prefix condition, and has an average length \bar{R} that satisfies the inequality $H(X) \leq \bar{R} < H(X) + 1$. Efficient representation of symbols leads to compression of data.
- The expected codeword length of Huffman code is bounded as $\bar{R} \leq H(X) + 1$.
- The expected codeword length of Shannon-Fano-Elias code is bounded as $\bar{R} < H(X) + 2$.
- The basic concept of Shannon-Fano-Elias coding is used in a computationally efficient algorithm for encoding and decoding called Arithmetic Coding which works by representing the file to be encoded by an interval of real numbers between 0 and 1.
- In contrast to the Huffman coding scheme, the Lempel-Ziv technique is independent of the source statistics. The Lempel-Ziv technique generates a fixed length code, whereas the Huffman code is a variable length code.
- Run-length Encoding, or RLE is a technique used to reduce the size of a repeating string of characters. This repeating string is called a run. Run-length encoding is supported by most bitmap file formats such as TIFF, BMP and PCX.
- Distortion implies some measure of difference between the actual source samples $\{x_k\}$ and the corresponding quantized value $\{\tilde{x}_k\}$. The squared-error distortion is given by $d(x_k, \tilde{x}_k) = (x_k - \tilde{x}_k)^2$. In general a distortion measure may be represented as $d(x_k, \tilde{x}_k) = |x_k - \tilde{x}_k|^p$.
- The Hamming distortion is defined as $d(x_k, \tilde{x}_k) = \begin{cases} 0 & \text{if } x_k = \tilde{x}_k \\ 1 & \text{if } x_k \neq \tilde{x}_k \end{cases}$.
- The minimum rate (in bits/source output) required to represent the output X of the memoryless source with a distortion less than or equal to D is called the rate distortion function $R(D)$, defined as $R(D) = \min_{p(\tilde{x}|x): E[d(X, \tilde{X})] \leq D} I(X, \tilde{X})$, where $I(X, \tilde{X})$ is the average mutual information between X and \tilde{X} .
- Some properties of rate distortion function: $R(D)$ is non-increasing in D , $R(D)$ is convex, $R(0) \leq H(X)$ and $R(D) = 0$ for $D \geq D_{max}$.
- The distortion resulting due to the quantization can be expressed as $D = \int_{-\infty}^{\infty} f(\tilde{x} - x)p(x)dx$, where $f(\tilde{x} - x)$ is the desired function of the error. An optimum quantizer is one that minimizes D by optimally selecting the output levels and the corresponding input range of each output level. The resulting optimum quantizer is called the Lloyd-Max quantizer.
- A discrete stochastic process X_1, X_2, \dots is said to be a Markov chain or a Markov process if, for $n = 1, 2 \dots$
- $$P(X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, \dots, X_1 = x_1) = P(X_{n+1} = x_{n+1} | X_n = x_n)$$
- for all $x_1, x_2, \dots x_n, x_{n+1} \in X$.

- The entropy rate of a stochastic process X is given by $H(X) = \lim_{n \rightarrow \infty} \frac{1}{n} H(X_1, X_2, \dots, X_n)$, provided the limit exists.
- For stationary Markov chain, the entropy rate is given by
$$H(X) = \lim_{n \rightarrow \infty} H(X_n | X_{n-1}, \dots, X_1) = \lim_{n \rightarrow \infty} H(X_n | X_{n-1}) = H(X_2 | X_1) \cdot$$
- Quantization and source coding techniques (Huffman coding, arithmetic coding and run-length coding) are used in the JPEG standard for image compression.
- MPEG stands for Moving Picture Coding Experts Group and encompasses a whole family of international standards for the compression of audio and video digital data.
- The important steps in MPEG compression are reduction of the resolution, motion estimation/compensation, Discrete Cosine Transformation (DCT), quantization and entropy coding.

Obstacles are those frightful things you see when you take your eyes off your goal.

Henry Ford (1863-1947)



MULTIPLE CHOICE QUESTIONS

1.1 Pick the correct choice

- | | |
|-------------------------------|-------------------------------|
| (a) $I(X; Y) = H(X) - H(X Y)$ | (b) $I(X; Y) = H(Y) - H(Y X)$ |
| (c) Both a and b | (d) None |

1.2 Which among the following is used to construct the binary code that satisfies the prefix condition?

- | | |
|----------------------------|-----------------------|
| (a) Information Rate | (b) Noiseless Channel |
| (c) Channel Coding Theorem | (d) Kraft Inequality |

1.3 The efficiency of a prefix code is given by

- | | |
|---------------------------------------|-----------------------------------|
| (a) $\eta = \frac{\bar{R}}{H(X)}$ | (b) $\eta = \frac{H(X)}{\bar{R}}$ |
| (c) $\eta = \frac{H(X) + 1}{\bar{R}}$ | (d) None of the above |

1.4 The Kullback Leibler (KL) distance violates which property of a distance measure

- | | |
|-------------------------|-----------------------|
| (a) Non-negative | (b) Symmetry property |
| (c) Triangle inequality | (d) None of the above |

- 1.5 The relationship between mutual information and joint entropy can be expressed as
 (a) $I(X; Y) = H(X) - H(Y) + H(X, Y)$ (b) $I(X; Y) = H(X) + H(Y) - H(X, Y)$
 (c) $I(X; Y) = H(X) + H(Y) + H(X, Y)$ (d) None of the above
- 1.6 The expected codeword length of Huffman code is bounded as
 (a) $\bar{R} \leq H(X) + 1$ (b) $\bar{R} \leq H(X) - 1$
 (c) $\bar{R} < H(X) + 1$ (d) None of the above
- 1.7 Which among the following compression techniques is/are intended for still images?
 (a) JPEG (b) H.263
 (c) MPEG (d) All of the above
- 1.8 Which coding technique(s) is a fixed length code?
 (a) Lempel Ziv (b) Huffman
 (c) Both a and b (d) None of the above
- 1.9 JPEG standard for image compression uses
 (a) Huffman coding (b) Arithmetic coding
 (c) Run-length coding (d) All of the above
- 1.10 The MPEG-standard reduces temporal redundancy by using
 (a) I-frames, P-frames and B-frames (b) Discrete Cosine Transformation (DCT)
 (c) Quantization (d) Entropy coding

For interactive quiz with answers, scan the QR code given here

Or

Visit <http://qr-code.flipick.com/index.php/566>



SHORT-ANSWER TYPE QUESTIONS



- 1.1 Can differential entropy be negative? Explain.
- 1.2 We require infinite number of bits to represent a continuous random variable precisely. True or false?
- 1.3 If X and Y are independent, is it true that $H(X, Y) = H(X) + H(Y)$?
- 1.4 Let X , Y , and Z be discrete random variables with $I(X; Y) = 0$ and $I(X; Z) = 0$. Is $I(X; Y, Z) = 0$?
- 1.5 Does conditioning reduce entropy? Is $H(X|Y) \leq H(X)$? Is $h(X|Y) \leq h(X)$?
- 1.6 When is $H(X|Y) = H(X)$?
- 1.7 Is the function $\log(x)$ concave or convex?
- 1.8 Is the function $x\log(x)$ concave or convex?
- 1.9 What is the value of $I(X; X)$?
- 1.10 Can mutual information be negative?
- 1.11 Is $\{1, 01, 001, 000\}$ a prefix code? What about $\{00, 01, 10, 11\}$?
- 1.12 Does the addition of independent random variables increase uncertainty?
- 1.13 By what factor does one bit reduce the expected distortion.
- 1.14 You have 12 identical gold coins of which one is a counterfeit (weighs either more or less than others). You have a beam balance. What is the minimum number of weighings needed to identify the counterfeit coin? How is it related to information theory?

1.15 Why is DCT used for JPEG compression rather than FFT?

For answers, scan the QR code given here

Or

Visit <http://qrcode.flipick.com/index.php/581>



PROBLEMS

- M** 1.1 Prove that the entropy for a discrete source is a maximum when the output symbols are equally probable.
- S** 1.2 Prove the inequality $\ln x \leq x - 1$. Plot the curves $y_1 = \ln x$ and $y_2 = x - 1$ to demonstrate the validity of this inequality.
- M** 1.3 Show that $I(X; Y) \geq 0$. Under what condition does the equality hold?
- S** 1.4 Let X denotes the number of tosses required for a coin until the first tail appears.
- (i) Find the entropy, $H_f(X)$ if the coin is fair.
 - (ii) Next assume the coin to be unfair with p being the probability of getting a tail. Find the entropy, $H_u(X)$.
- S** 1.5 Consider a code that satisfies the suffix condition: No codeword is a suffix of any other codeword.
- (i) Is this a uniquely decodable code? Why?
 - (ii) Can you relate the minimum average length of suffix codes to the average length of the Huffman code for the same random variable? Explain.
- M** 1.6 Consider the Kullback Leibler distance between two probability mass functions $p(x)$ and $q(x)$, given by $D(p\|q) = \sum_{x \in X} p(x) \log\left(\frac{p(x)}{q(x)}\right)$.
- (i) Show that $D(p\|q)$ is non negative.
 - (ii) Show that the Kullback Leibler distance does not follow the other two properties of a distance, i.e., symmetry and the triangle inequality.
 - (iii) Show that $I(X; Y) = D(p(x,y)\|p(x)p(y))$. This would elegantly prove that mutual information is non negative.
- D** 1.7 Suppose a source produces *independent* symbols from the alphabet $\{a_1, a_2, a_3\}$, with probabilities $p_1 = 0.4999999$, $p_2 = 0.4999999$, and $p_3 = 0.0000002$.
- (i) Compute the entropy, $H(X)$, of this source.
 - (ii) Find an optimal code for this source, and compute its expected codeword length.
 - (iii) Find an optimal code for the second extension of this source (i.e., for blocks of two symbols), and compute its expected codeword length, and the expected codeword length divided by two.
 - (iv) Prove that in order to compress to within 1% of the entropy by encoding blocks of size N from this source, N will have to be at least 5. Note that if you use (1.45) you get a very loose upper bound.

- S** 1.8 Consider an integer values random variable, X , given by $P(X = n) = \frac{1}{An \log^2 n}$, where $A = \sum_{n=2}^{\infty} \frac{1}{n \log^2 n}$ and $n = 2, 3, \dots, \infty$. Find the entropy, $H(X)$.
- S** 1.9 Let us see what happens if $p(x_i)$ in (1.14) is replaced by $p(x_i)\Delta x_i$. Show that the expression for entropy for discrete random variables does not converge to (1.30) and can be simplified to
- $$H(X) = - \int_{-\infty}^{\infty} p(x) \log p(x) - \lim_{\Delta x \rightarrow 0} \log(\Delta x)$$
- M** 1.10 Find the differential entropy of the random variable $Z = X_1 + X_2$, where X_1 and X_2 are independent Gaussian random variables with means μ_i and variances σ_i^2 , $i = 1, 2$.
- S** 1.11 Consider a DMS with source probabilities $\{0.20, 0.20, 0.15, 0.15, 0.10, 0.10, 0.05, 0.05\}$.
- (i) Determine an efficient fixed length code for the source.
 - (ii) Determine the Huffman code for this source.
 - (iii) Compare the two codes and comment.
- M** 1.12 A DMS has three output symbols with probabilities $\{0.5, 0.4, 0.1\}$.
- (i) Determine the Huffman code for this source and find the efficiency η .
 - (ii) Determine the Huffman code for this source taking *two* symbols at a time and find the efficiency η .
 - (iii) Determine the Huffman code for this source taking *three* symbols at a time and find the efficiency η .
- S** 1.13 For a source with entropy $H(X)$, prove that the entropy of a B -symbol block is $BH(X)$.
- M** 1.14 Let the probability of getting a head for a biased coin be $p = \left(\frac{1}{x^{x^{-1}}} \right)$ and the probability of getting a tail be $q = \left(\frac{1}{y^{y^{-1}}} \right)$.
- (i) Show that the entropy of this biased coin is $\log_2(xy)$.
 - (ii) Give the plot of x versus p .
- M** 1.15 Let X and Y be random variables that take on values x_1, x_2, \dots, x_r and y_1, y_2, \dots, y_s respectively. Let $Z = X + Y$.
- (i) Show that $H(Z|X) = H(Y|X)$.
 - (ii) If X and Y are independent, then argue that $H(Y) \leq H(Z)$ and $H(X) \leq H(Z)$. Comment on this observation.
 - (iii) Find an example of X and Y with $H(X) > H(Z)$ and $H(Y) > H(Z)$.
 - (iv) Under what condition will $H(Z) = H(X) + H(Y)$?
- S** 1.16 Let X be a discrete random variable and let $Y = f(X)$ be a deterministic function of X .
- (i) Show that $H(X) = H(Y) + H(X|Y)$.
 - (ii) Under what condition is $H(X) = H(Y)$?
- S** 1.17 (i) Show that the entropy of a random variable X is zero if and only if X has only one value with non-zero probability.
- (ii) Show that if the conditional entropy $H(X|Y)$ is zero, then the value of X is completely determined by the value of Y , with probability 1.

- S** 1.18 Can we construct a ternary Huffman code with codeword lengths 2, 2, 2, 2, 2, 2, 2, 2, 3, 3 and 3? Explain.
- S** 1.19 Consider the variable length code $C_1 = \{00, 01, 0\}$. Is it uniquely decodable? Is it an instantaneous code? What about the variable length code $C_2 = \{00, 01, 100, 101, 11\}$? Is it uniquely decodable? Instantaneous?
- M** 1.20 In a red colored urn, I have 1 violet colored ball, 2 indigo colored balls, 3 blue colored balls, 4 green colored balls, 5 yellow colored balls and 6 orange colored balls. From this urn, I intend to pick up a ball at random and indicate the color of the ball using an efficient code. Suggest a (a) binary code and (b) ternary code for doing so. What is the expected codeword length in each case?
- M** 1.21 We know that in Huffman coding, the most probable symbol gets assigned the shortest codeword length. Practically, the shortest codeword length possible is 1. Show that, for the most probable symbol x_1 , if the associated probability $p_1 > 0.4$, then the codeword will necessarily be of length 1.
- D** 1.22 Consider Lempel Ziv encoding for quaternary data (symbols: 0, 1, 2, 3).
 - (i) Encode the following quaternary data: 1 3 3 0 0 2 0 2 1 1 1 3 0 0 0 0 2 2 1 2 2 2 3 3. What is the compression ratio obtained, where the compression ratio is defined as the number of bits after encoding divided by the number of bits before encoding.
 - (ii) Next, represent each quaternary symbol by its binary equivalent (0 → 00, 1 → 01, 2 → 10, 3 → 11). Now perform Lempel Ziv encoding on the binary data and obtain the compression ratio.
 - (iii) Compare the results of (i) and (ii). Comment.
- D** 1.23 Consider a bit stream being generated by source A. The probabilities of occurrence of ‘runs’ in the bit stream is given by

$$p_r = \begin{cases} 2^{-r} & \text{for } 1 \leq r < n \\ 2^{-(n-1)} & \text{for } r = n \end{cases} \quad (1.81)$$

where r is the length of the run and n is the maximum run length possible. Note that the probabilities correspond to the run length, irrespective of whether it’s a run of 1’s or 0’s.

- (i) Design a run length code and find out the compression it will provide.
 - (ii) Suppose you want to encode the runs using Huffman code. Generate the code for this bit stream.
- What is the compression provided by this code?

- M** 1.24 We next want to try a new scheme for run length coding. We repeat the run-length encoding process again and again until no more compression is possible. Using this procedure, if we have to encode a run of, say n 1’s, what will be the maximum possible (best case) compression?
- M** 1.25 Consider a source X uniformly distributed on the set $\{1, 2, \dots, m\}$. Find the rate distortion function for this source with Hamming distortion defined as $d(x, \tilde{x}) = \begin{cases} 0, & x = \tilde{x} \\ 1, & x \neq \tilde{x} \end{cases}$.

COMPUTER PROBLEMS



- 1.1 Write a program that performs Huffman coding, given the source probabilities. It should generate the code and give the coding efficiency. It should be able to handle non-binary inputs as well.
 - 1.2 Modify the above program so that it can group together n source symbols and then generate the Huffman code. Plot the coding efficiency η versus n for the following source symbol probabilities: $\{0.55, 0.25, 0.20\}$. For what value of n does the efficiency become better than 0.9999? Repeat the exercise for the following source symbol probabilities $\{0.45, 0.25, 0.15, 0.10, 0.05\}$.
 - 1.3 Write a program that executes the Lempel-Ziv algorithm. The input to the program can be the English alphabets. It should convert the alphabets to their ASCII code and then perform the compression routine. It should output the compression achieved. Using this program, find out the compression achieved for the following strings of letters:
 - (i) The Lempel-Ziv algorithm can compress the English text by about fifty five percent.
 - (ii) The cat can not sit on the canopy of the car.
 - 1.4 Write a program that performs Run Length Encoding (RLE) on a sequence of bits and gives the coded output along with the compression ratio. What is the output of the program if the following sequence is fed into it:

1100000000111000001111111111111100000110000000

Now feed back the *encoded output* to the program, i.e., perform the RLE two times on the original sequence of bits. What do you observe? Comment on your observation.

- 1.5 Suppose for a certain type of communication we only use three letters A, B and C. The probability of occurrences for these are $\{0.5, 0.3, 0.2\}$. However, the use of these letters is not independent. The probability of occurrences of the bigrams are $P(AB) = 0.25$, $P(BC) = 0.05$, $P(CA) = 0.35$, $P(BA) = 0.15$, $P(CB) = 0.05$, $P(AC) = 0.15$. Let us call the Huffman code which takes into consideration the bigrams as Pseudo Markov Huffman (PMH) code.

 - (i) Design a PMH code for this communication scenario which takes into consideration the bigrams as well. Write a computer program that can take these probabilities as input and generate the PMH code.
 - (ii) Write a program that performs arithmetic coding for this case and compare with the result of the PMH code.

1.6 Write a program that takes in a 2^n level gray scale image (n bits per pixel) and performs the following operations:

 - (i) Breaks it up into 8 by 8 pixel blocks.
 - (ii) Performs DCT on *each* of the 8 by 8 blocks.
 - (iii) Quantizes the DCT coefficients by retaining only the m Most Significant Bits (MSB), where $m \leq n$.
 - (iv) Performs the zig-zag coding followed by run length coding.
 - (v) Performs Huffman coding on the bit stream obtained above (think of a reasonable way of calculating the symbol probabilities).
 - (vi) Calculates the compression ratio.
 - (vii) Performs the decompression (*i.e.*, the inverse operation of the steps (v) back to (i)).

Perform image compression using this program for different values of m . Up to what value of m is there no perceptible difference in the original image and the compressed image?

PROJECT IDEAS

1.1 Superinformation: We wish to determine the ‘entropy of entropy’, which we define as *superinformation*.

- (i) Generate a ‘long string’ of characters randomly. The number of characters and their probabilities of occurrences are design parameters.
 - (ii) Divide the string into sub-blocks of size B .
 - (iii) For i^{th} sub-block, find the entropy, H_i .
 - (iv) Find the histogram of H_i . Normalize this histogram to obtain a pdf.
 - (v) Find the entropy for this pdf. This is the superinformation (entropy of entropy) of the sequence.
 - (vi) Use the concept of superinformation on DNA sequences and see if you can segregate exons (coding regions) from introns (non-coding regions).

1.2 Efficiency of the Morse Code: The Morse Code for the English alphabet is given in Fig. 1.21.

- (i) Determine the average codeword length for the Morse Code and determine the efficiency of the code.
 - (ii) Is it uniquely decodable? Explain.
 - (iii) Compare Morse code with the Huffman code for the English alphabet.

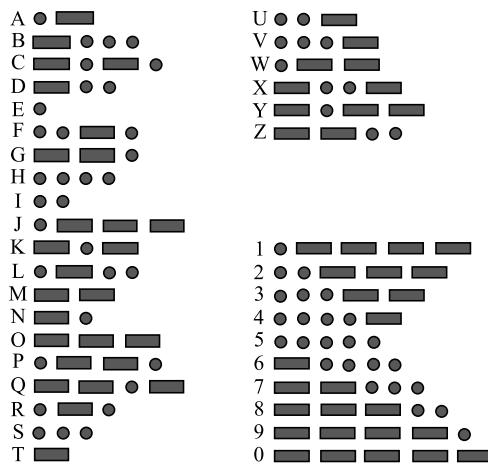


Fig. 1.21

1.3 Entropy-based Image Partitioning: Consider an energy-limited wireless device that periodically captures images and uploads it over a wireless channel. Energy is expended in the device for wireless transmission of bits (E_t , $\mu\text{J}/\text{bit}$) and computation (E_c , $\mu\text{J}/\text{bit}$). We would like to optimize the total energy per bit. We have three options (i) We upload the uncompressed image (i.e., we have to transmit a larger number of bits). (ii) We compress the entire image and then send (we will save energy by transmitting a shorted bit stream but spend energy doing compression). (iii) We can compress part of the image (based on entropy-based partitioning) and send partly compressed and partly uncompressed (raw bits). You can first divide the image into blocks and see which ones to compress. Design an optimal scheme using entropy-based image partitioning in order to minimize energy per bit. Also carry out simulations using realistic numbers for E_t (say, for IEEE 802.11x) and E_c (say, for the ARM processor).

1.4 High Efficiency Video Coding (HEVC): Consider the HEVC standard, which is a successor to the Advanced Video Coding standard (H.264/MPEG-4). HEVC uses a context-adaptive binary arithmetic coding (CABAC) algorithm. CABAC has multiple probability modes for different contexts. It works on binary data by first converting all non-binary symbols to binary. Then, for each bit, the coder selects an appropriate probability model to use. It then uses information from nearby elements to optimize the probability estimates. Finally, arithmetic coding is used to compress the data. Develop a simplified HEVC scheme using CABAC and perform encoding on simple binary images. Comment on your results.

REFERENCES FOR FURTHER READING

1. Cover, Thomas M. and Joy, A. Thomas; *Elements of Information Theory*, John Wiley & Sons, 2012.
2. Robert, Ash B. *Information Theory*. Dover Special Priced Titles, 2007.

Channel Capacity and Coding

Experimentalists think that it is a mathematical theorem while the mathematicians believe it to be an experimental fact.

[On the Gaussian curve, remarked to Poincaré:]

Lippman, Gabriel (1845-1921)

Learning objectives

After studying this chapter the students will be able to:

- LO 1** Categorise different types of channels and channel models.
- LO 2** Determine channel capacity mathematically and the capacity of a given channel.
- LO 3** Explain the need for channel coding in the context of the Noisy Channel Coding Theorem.
- LO 4** State and prove the Information Capacity Theorem and discuss its ramifications.
- LO 5** Discuss random selection of codes and define the cutoff rate.

2.1 Introduction

In the previous chapter we saw that most of the natural sources have inherent redundancies and it is possible to compress data by removing these redundancies. Compression is possible by different source coding techniques. After efficient representation of source symbols by the minimum possible number of bits, we need to transmit these bit-streams over channels (e.g., telephone lines, optical fibres, wireless channels, etc.). These bits may be transmitted as they are (for baseband communications), or after modulation (for passband communications). Unfortunately, all real-life channels are noisy (this is not so unfortunate for those who make a living out of designing communication systems for noisy channels!). The term noise is used to designate unwanted waves that tend to disturb the transmission and processing of the wanted signals in

...
This chapter comes with a video overview by the author. Scan here to know more or
Visit <http://qrcode.flipick.com/index.php/589>



communication systems. The sources of noise may be external to the system (e.g., atmospheric noise, man generated noise etc.), or internal to the system (e.g., thermal noise, shot noise etc.). In effect, the bit stream obtained at the receiver end is likely to be different from the bit stream that is transmitted. For the case of passband communication, the demodulator processes the channel-corrupted waveform and reduces each waveform to a scalar or a vector that represents an estimate of the transmitted data symbols. The detector, which follows the demodulator, may decide on whether the transmitted bit is a 0 or a 1. This is called a **hard decision decoding**. This decision process at the decoder is like a binary quantization with two levels. If there are more than 2 levels of quantization, the detector is said to perform a **soft decision decoding**. In the extreme case, no quantization is performed for soft decision decoding.

The use of hard decision decoding causes an irreversible loss of information at the receiver. Suppose the modulator sends only binary symbols but the demodulator has an alphabet with Q symbols. Assuming the use of the quantizer as depicted in Fig. 2.1 (a), we have $Q = 8$. Such a channel is called a **Binary input Q -ary output Discrete Memoryless Channel**. The corresponding channel is shown in Fig. 2.1 (b). The decoder performance depends on the location of the representation levels of the quantizers, which in turn depends on the signal level and the noise power. Accordingly, the demodulator must incorporate automatic gain control in order to realize an effective multilevel quantizer. It is clear that the construction of such a decoder is more complicated than the hard decision decoder. However, soft decision decoding can provide significant improvement in the performance over hard decision decoding.

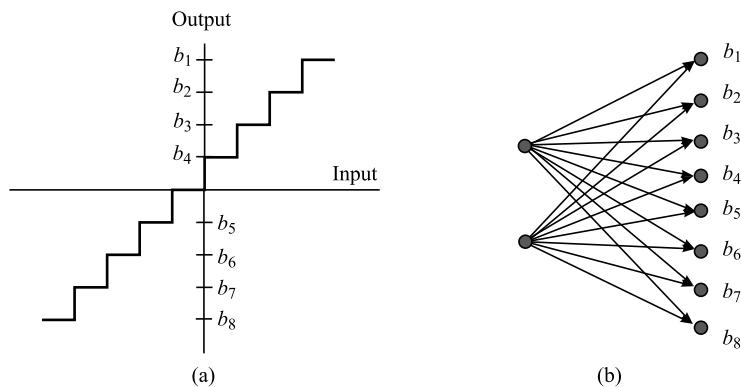


Fig. 2.1 (a) Transfer characteristic of multilevel quantizer. (b) binary input Q -ary output discrete memoryless channel.

There are three balls that a digital communication engineer must juggle: (i) the transmitted signal power (ii) the channel bandwidth and (iii) the reliability of the communication system (in terms of the bit error rate). Channel coding allows us to trade-off one of these commodities (signal power, bandwidth or reliability) with respect to others. In this chapter, we will study how to achieve reliable communication in the presence of noise. We shall ask ourselves questions like how many bits per second can be sent reliably over a channel of a given bandwidth and for a given signal to noise ratio (SNR)? For that, we begin by studying a few channel models first.

In this chapter

We will start with defining the different types of channel models such as the Binary Symmetric Channel, which is a special case of the Discrete-input, Discrete-output channel, the Discrete Memoryless Channel, the Multiple Input Multiple Output channel, the Relay Channel, Multiple Access Channel and the Broadcast Channel, in LO 1.

Next, we will define channel capacity and learn how to calculate the capacity of a given channel. We will look at specific examples of the Binary Symmetric Channel, Binary Erasure Channel and Weakly Symmetric Channels, through LO 2.

We will then motivate the need for Channel Coding and look at the Noisy Channel Coding Theorem. We will also discuss what is meant by the achievability of a rate. All this in LO 3.

Next, we will state and prove the Information Capacity Theorem, and discuss reliable communication over unreliable channels. We will apply this knowledge to study parallel Gaussian channels, the capacity of Multiple Input Multiple Output channels and the capacity region for Multiple Access Channels, in LO 4.

Finally, in LO 5, we will discuss random selection of codes and define the Cutoff Rate.

2.2 Channel Models

LO 1



Categorise different types of channels and channel models.

We have already come across the simplest of the channel models, the **Binary Symmetric Channel** (BSC), in the previous chapter. If the modulator employs binary waveforms and the detector makes hard decisions, then the channel may be viewed as one in which a binary bit stream enters at the transmitting end and another bit stream comes out at the receiving end. A BSC is shown in Fig. 2.2 (a). Figure 2.2(b) shows one possible output for a binary image sent through a given BSC with $p = 0.1$.

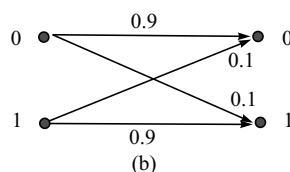
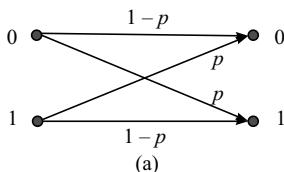


Fig. 2.2 (a) A binary symmetric channel (BSC). (b) How an image might look after transmission through a BSC.

This *binary Discrete-input, Discrete-output channel* is characterized by the set $X = \{0,1\}$ of possible inputs, the set $Y = \{0,1\}$ of possible outputs and a set of conditional probabilities that relate the possible outputs to the possible inputs. Let the noise in the channel cause independent errors in the transmitted binary sequence with average probability of error p .

Then,

$$\begin{aligned} P(Y=0|X=1) &= P(Y=1|X=0) = p \\ P(Y=1|X=1) &= P(Y=0|X=0) = 1-p \end{aligned} \tag{2.1}$$

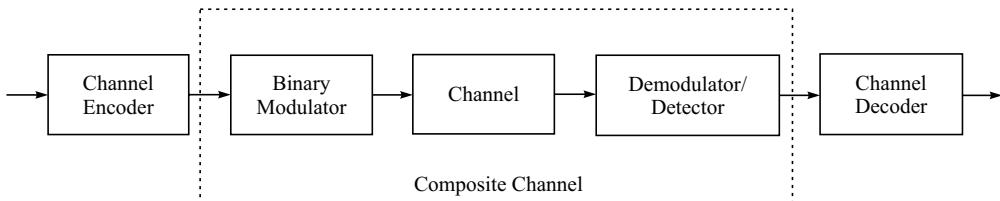


Fig. 2.3 A composite discrete-input, discrete-output channel formed by including the modulator and demodulator/detector.

The BSC is a special case of a **composite Discrete-input, Discrete-output channel** depicted in Fig. 2.3. Let the input to the channel be q -ary symbols, i.e., $X = \{x_0, x_1, \dots, x_{q-1}\}$ and the output of the detector at the receiving end of the channel consist of Q -ary symbols, i.e., $Y = \{y_0, y_1, \dots, y_{Q-1}\}$. We assume that the channel and the modulation are both memoryless. The inputs and the outputs can then be related by a set of qQ **conditional probabilities**

$$P(Y=y_i | X=x_j) = P(y_i | x_j) \quad (2.2)$$

where $i = 0, 1, \dots, Q - 1$ and $j = 0, 1, \dots, q - 1$. This channel is known as a **Discrete Memoryless Channel** (DMC) and is depicted in Fig. 2.4.

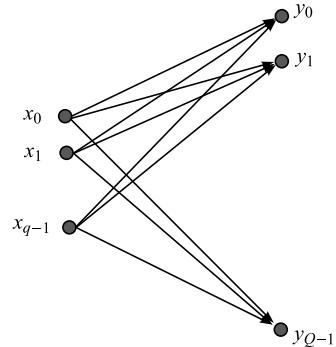


Fig. 2.4 A discrete memoryless channel (DMC) with q -ary input and Q -ary output.



Definition 2.1 The conditional probability $P(y_i | x_j)$ is defined as the **Channel Transition Probability** and is denoted by p_{ji} .

Definition 2.2 The conditional probabilities $\{P(y_i | x_j)\}$ that characterize a DMC can be arranged in the matrix form $\mathbf{P} = [p_{ji}]$. \mathbf{P} is called the **Probability Transition Matrix** for the channel. Note: Sometimes, \mathbf{P} is also referred to as the **Transition Probability Matrix** or the **Channel Transition Matrix** or the **Channel Transition Probability Matrix**.

So far we have discussed a single channel with discrete inputs and discrete outputs. We can decouple the modulator and the demodulator from the physical channel (as depicted in Fig. 2.3). Thus, the input to the channel will be a waveform and the output of the channel will also be a waveform. These are called **Waveform Channels**. Such channels are typically associated with a given bandwidth, W . Suppose, $x(t)$ is a bandlimited input to a waveform channel and $y(t)$ is the output of the channel. Then,

$$y(t) = x(t) + n(t) \quad (2.3)$$

where, $n(t)$ is the additive noise.

It is also possible to realize multiple channels between the transmitter and receiver. Such channels can be readily realized in wireless communication scenarios by using multiple antennas at the transmitter and receiver. The four obvious combinations are as follows:

- (i) **Single Input Single Output** (SISO): This refers to the familiar wireless configuration with a single antenna both at the transmitter and receiver.

- (ii) **Single Input Multiple Output (SIMO)**: This refers to the configuration with a single antenna at the transmitter and multiple antennas at the receiver.
- (iii) **Multiple Input Single Output (MISO)**: This refers to the configuration with multiple antennas at the transmitter but only a single antenna at the receiver.
- (iv) **Multiple Input Multiple Output (MIMO)**: This refers to the most general configuration with multiple antennas both at the transmitter and receiver.

Consider an MIMO system with M_T transmits antennas and M_R receive antennas. Let us denote the impulse response between the j^{th} ($j = 1, 2, \dots, M_T$) transmit antenna and the i^{th} ($i = 1, 2, \dots, M_R$) receive antenna by $h_{ij}(\tau, t)$. Note that here we are considering the waveform channel and the modulator/demodulator is not a part of the channel. The MIMO channel can be represented using a $M_R \times M_T$ matrix as follow.

$$\mathbf{H}(\tau, t) = \begin{bmatrix} h_{1,1}(\tau, t) & h_{1,2}(\tau, t) & \cdots & h_{1,M_T}(\tau, t) \\ h_{2,1}(\tau, t) & h_{2,2}(\tau, t) & \cdots & h_{2,M_T}(\tau, t) \\ \vdots & \vdots & \ddots & \vdots \\ h_{M_R,1}(\tau, t) & h_{M_R,2}(\tau, t) & \cdots & h_{M_R,M_T}(\tau, t) \end{bmatrix} \quad (2.4)$$

A wireless channel is time varying and the variable τ is used to capture the time-varying nature of the channel. If a signal $s_j(t)$ is transmitted from the j^{th} transmit antenna, the signal received at the i^{th} receive antenna is given by

$$y_i(t) = \sum_{j=1}^{M_T} h_{i,j}(\tau, t) s_j(t), \quad i = 1, 2, \dots, M_R \quad (2.5)$$

The input-output relation of a MIMO channel can be expressed succinctly in matrix notation as

$$\mathbf{y}(t) = \mathbf{H}(\tau, t) \mathbf{s}(t) \quad (2.6)$$

where, $\mathbf{s}(t) = [s_1(t) \ s_2(t) \ \cdots \ s_{M_T}(t)]^T$ and, $\mathbf{y}(t) = [y_1(t) \ y_2(t) \ \cdots \ y_{M_R}(t)]^T$



Each link between a pair of transmitter and receiver can be independently represented as a discrete memoryless channel.

DID YOU KNOW ?

We can also have **Relay Channels** where there is a source, a destination and intermediate relay nodes. These relay nodes facilitate the communication between the source and the destination, as shown in Fig. 2.5(a). There could be several ways to facilitate the transfer of information from the source to the destination by hopping over the intermediate nodes. One possibility is that each relay node simply amplifies the received signal and forwards it to the next relay node, maintaining a fixed average transmit power. This protocol is known as **Amplify-and-Forward (AF)** scheme. Alternately, the relay node can first decode the received signal and then re-encodes the signal before forwarding it to the next relay node. The processing of the signal at each relay node requires making a hard decision. This protocol is known as **Decode-and-Forward (DF)** scheme.

So far we have considered only a single transmitter and a single receiver (though, each one of the transmitters or receivers can have multiple antennas!). In real-life, there can be multiple senders and multiple receivers (with each one having the possibility of multiple antennas!). Suppose M transmitters (say, mobile phone users) want to communicate with a single receiver (say, the base station) over a common channel, as depicted in Fig. 2.5 (b). This scenario is known as a **Multiple Access Channel**. We can also reverse the scenario. Suppose a single transmitter (say, a low earth orbit satellite) wants to communicate with M

receivers (say, the home dish antennas) over a common channel, as shown in Fig. 2.5 (c). This is an example of a **Broadcast Channel**.

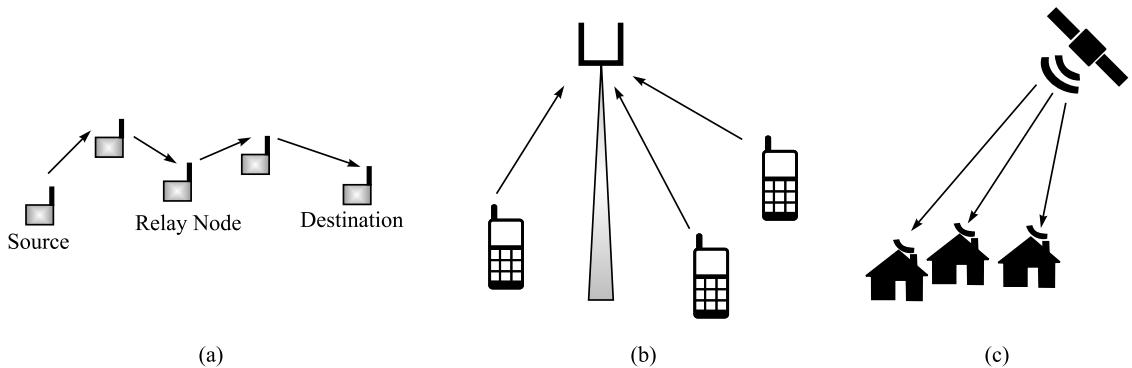


Fig. 2.5 (a) Relay Channel. (b) Multiple Access Channel. (c) Broadcast Channel.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 1:

1. We would like to represent the wireless local area home network consisting of one access point and three laptops simultaneously connected to it. What types of channel model should we use? S
2. Graphically represent a binary channel with $P(Y=0|X=1)=p$, $P(Y=1|X=0)=q$, $P(Y=1|X=1)=1-p$, $P(Y=0|X=0)=1-q$. S
3. Consider the binary channel shown in Fig. 2.6. Let the *a priori* probabilities of sending the binary symbol be p_0 and p_1 , where $p_0 + p_1 = 1$. Find the *a posteriori* probabilities M

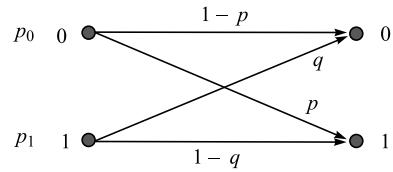


Fig. 2.6

4. Show that a cascade of n BSCs can be effectively represented as a single BSC. M

If you have successfully solved the above problems,
you have mastered LO 1. Congratulations!



The answers are available here. Scan to check
Or
Visit <http://qrcode.flipick.com/index.php/583>



Levels of Difficulty

- S **Simple:** Level 1 and Level 2 Category
- M **Medium:** Level 3 and Level 4 Category
- D **Difficult:** Level 5 and Level 6 Category

2.3 Channel Capacity

LO 2



Determine channel capacity mathematically and the capacity of a given channel.

Consider a DMC having an input alphabet $X = \{x_0, x_1, \dots, x_{q-1}\}$ and an output alphabet $Y = \{y_0, y_1, \dots, y_{r-1}\}$. Let us denote the set of channel transition probabilities by $P(y_i|x_j)$. The average mutual information provided by the output Y about the input X is given by (see Chapter 1, Section 1.3)

$$I(X; Y) = \sum_{j=0}^{q-1} \sum_{i=0}^{r-1} P(x_j) P(y_i | x_j) \log \frac{P(y_i | x_j)}{P(y_i)} \quad (2.7)$$

The channel transition probabilities $P(y_i|x_j)$ are determined by the channel characteristics (primarily the noise in the channel). However, the input symbol probabilities $P(x_j)$ are within the control of the discrete channel encoder. The value of the average mutual information, $I(X; Y)$, maximized over the set of input symbol probabilities $P(x_j)$ is a quantity that depends only on the channel transition probabilities $P(y_i|x_j)$ (hence only on the characteristics of the channel). This quantity is called the **Capacity of the Channel**.

 **Definition 2.3** The **Capacity** of a discrete memoryless channel (DMC) is defined as the maximum average mutual information in any single use of the channel, where the maximization is over all possible input probabilities. That is,

$$\begin{aligned} C &= \max_{P(x_j)} I(X; Y) \\ &= \max_{P(x_j)} \sum_{j=0}^{q-1} \sum_{i=0}^{r-1} P(x_j) P(y_i | x_j) \log \frac{P(y_i | x_j)}{P(y_i)} \end{aligned} \quad (2.8)$$

The maximization of $I(X; Y)$ is performed under the constraints

$$P(x_j) \geq 0 \quad \text{and} \quad \sum_{j=0}^{q-1} P(x_j) = 1$$

 The units of channel capacity is bits per channel use (provided the base of the logarithm is 2). If the base of the logarithm is e , the units of capacity is in nats per channel use (coming from the use of natural logarithm).

Example 2.1 Consider a BSC with channel transition probabilities

$$P(0|1) = p = P(1|0)$$

Thus the transition probability matrix is given by $\mathbf{P} = \begin{bmatrix} 1-p & p \\ p & 1-p \end{bmatrix}$, and p is referred to as the **crossover probability**.

By symmetry, the capacity, $C = \max_{P(x_j)} I(X;Y)$, is achieved for input probabilities $P(0) = 0.5 = P(1)$. From (2.8) we obtain the capacity of a BSC as

$$C = 1 + p \log_2 p + (1-p) \log_2 (1-p)$$

Let us define the **entropy function**

$$H(p) = -p \log_2 p - (1-p) \log_2 (1-p)$$

Hence, we can rewrite the capacity of a binary symmetric channel as

$$C = 1 - H(p)$$

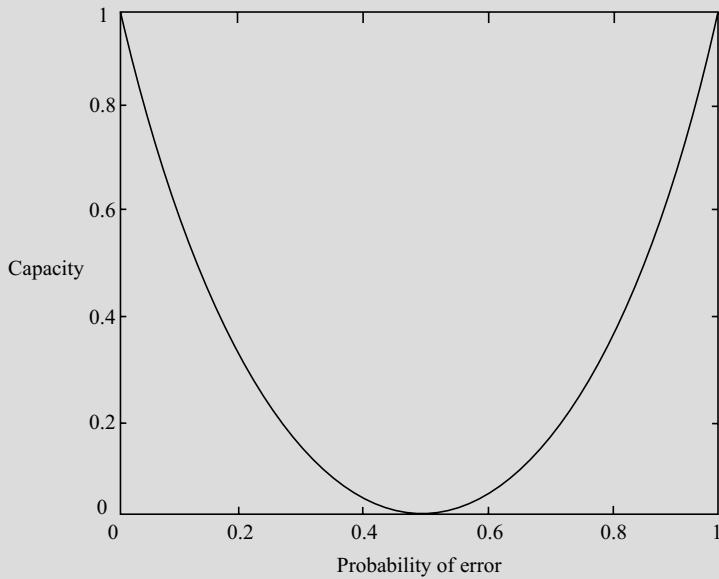


Fig. 2.7 The capacity of a BSC.

The plot of the capacity versus p is given in Fig. 2.7. From the plot we make the following observations:

- (i) For $p = 0$ (i.e., noise free channel), the capacity is 1 bit/use, as expected. Each time we use the channel, we can successfully transmit 1 bit of information.
- (ii) For $p = 0.5$, the channel capacity is 0, i.e., observing the output gives no information about the input. It is equivalent to the case when the channel is broken. We can, might as well, discard the channel and toss a fair coin in order to estimate what was transmitted.
- (iii) For $0.5 < p < 1$, the capacity increases with increasing p . In this case, we simply reverse the positions of 1 and 0 at the output of the BSC.
- (iv) For $p = 1$ (i.e., every bit get flipped by the channel), the capacity is again 1 bit/use, as expected. In this case, one simply flips the bit at the output of the receiver so as to undo the effect of the channel.
- (v) Since p is a monotonically decreasing function of SNR, the capacity of a BSC is a monotonically increasing function of SNR.

Example 2.2 Consider a Binary Erasure Channel (BEC), shown in Fig. 2.8 with transition probability matrix given by $P = \begin{bmatrix} 1-\varepsilon & \varepsilon & 0 \\ 0 & \varepsilon & 1-\varepsilon \end{bmatrix}$. We will calculate the capacity of this BEC.

$$C = \max_{P(x_j)} I(X;Y) = \max_{P(x_j)} (H(Y) - H(Y|X)) = \max_{P(x_j)} (H(Y) - H(\varepsilon))$$

Note that given the input X , there are two possible outcomes (Y) with probabilities $\{1 - \varepsilon, \varepsilon\}$. Hence, $H(Y|X) = H(\varepsilon)$. For the output Y , we have three cases with $P_Y(0) = p(1 - \varepsilon)$, $P_Y(\varepsilon) = \varepsilon$ and $P_Y(1) = (1 - p)(1 - \varepsilon)$. Therefore,

$$H(Y) = H(p(1 - \varepsilon), \varepsilon, (1 - p)(1 - \varepsilon)) = H(\varepsilon) + (1 - \varepsilon) H(p)$$

Hence,

$$C = \max_{P(x_j)} (H(Y) - H(\varepsilon)) = \max_p ((1 - \varepsilon)H(p) + H(\varepsilon) - H(\varepsilon)) = \max_p ((1 - \varepsilon)H(p))$$

We know that the maximum value of $H(p) = 1$ for $p = 0.5$. Therefore, the capacity of a BEC is

$$C_{\text{BEC}} = 1 - \varepsilon$$

This result is very intuitive. Suppose we use the channel n times, i.e., we attempt to send n bits through the channel. Of these, $n\varepsilon$ bits are ‘lost’(erased) in the channel. So, we can recover, at most, $(1 - \varepsilon)n$ bits. Hence, per channel use, we can successfully send $(1 - \varepsilon)$ bits, leading to $C_{\text{BEC}} = 1 - \varepsilon$.

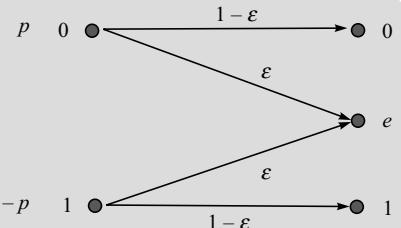


Fig. 2.8 The binary erasure channel (BEC).



Definition 2.4 A discrete memoryless channel is said to be **Symmetric** if the rows of the channel transition probability matrix are permutations of each other and the columns are permutations of each other.

Definition 2.5 A discrete memoryless channel is said to be **Weakly Symmetric** if the rows of the channel transition probability matrix are permutations of each other and the column sums $\sum_{x_i} p(y|x_i)$ are equal. For weakly symmetric channels, the capacity is given by

$$C = \log|Y| - H(\text{row of transition matrix}) \quad (2.9)$$

and this is obtained for uniform distribution on the input alphabet. Here $|Y|$ represents the cardinality of Y .

Example 2.3

Consider the transition probability matrix given by $P_1 = \begin{bmatrix} 0.5 & 0.3 & 0.2 \\ 0.2 & 0.5 & 0.3 \\ 0.3 & 0.2 & 0.5 \end{bmatrix}$.

This represents a symmetric channel. It can be easily verified that the rows of the channel transition probability matrix are permutations of each other, and so are the columns. On the other hand, consider

$P_2 = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 6 & 3 \\ 1 & 1 & 1 \\ 6 & 2 & 3 \end{bmatrix}$. This is a weakly symmetric channel, but not symmetric. While the rows of the channel

transition probability matrix are permutations of each other, the columns are not. However, each column adds up to $2/3$. From (2.9), the capacity of this weakly symmetric channel is readily calculated to be

$$C = \log(3) - H\left(\frac{1}{2}, \frac{1}{6}, \frac{1}{3}\right) = 0.1258 \text{ bits/use}$$

Properties

We now list some properties of channel capacity.

1. $C \geq 0$, since $I(X; Y) \geq 0$.
2. $C \leq \log|X|$, since $C = \max I(X; Y) \leq \max H(X) = \log|X|$.
3. $C \leq \log|Y|$, since $C = \max I(X; Y) \leq \max H(Y) = \log|Y|$.

Having developed the notion of capacity of a channel, we shall now try to relate it to reliable communication over the channel. So far, we have only talked about bits that can be sent over a channel each time it is used (bits/use). But, what is the number of bits that can be sent per second (bits/sec)? To answer this question we introduce the concept of **Channel Coding**.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 2:

- Find the capacity of the non-symmetric binary *erasure* channel shown in Fig. 2.9, where p_0 and p_1 are the *a priori* probabilities.
- Find the capacity of the channel shown in Fig. 2.10.

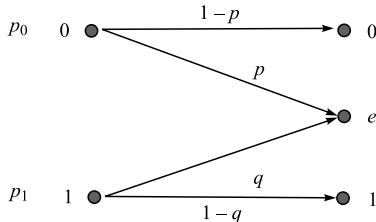


Fig. 2.9

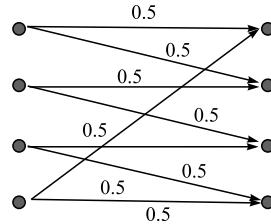


Fig. 2.10

- Determine the capacity of the channel given by $\mathbf{P} = \begin{bmatrix} \frac{1-p}{2} & \frac{1-p}{2} & \frac{p}{2} & \frac{p}{2} \\ \frac{p}{2} & \frac{p}{2} & \frac{1-p}{2} & \frac{1-p}{2} \end{bmatrix}$.

If you have successfully solved the above problems,
you have mastered LO 2. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/584>



2.4 Channel Coding

LO 3



Explain the need for channel coding in the context of the Noisy Channel Coding Theorem.

DID YOU KNOW ?

All real-life channels are affected by noise. Noise causes discrepancies (errors) between the input and the output data sequences of a digital communication system. For a typical noisy channel, the probability of bit error may be as high as 10^{-2} . This means that, on an average, 1 bit out of every 100 bits that are transmitted over this channel gets flipped. For most applications, this *level of reliability* is far from adequate. Different applications require different levels of

reliability (which is a component of the quality of service). Table 2.1 lists the typical acceptable bit error rates for various applications.

Table 2.1 Acceptable bit error rates for various applications

Application	Probability of Error
Speech telephony	10^{-4}
Voice band data	10^{-6}
Electronic mail, Electronic newspaper	10^{-6}
Internet access	10^{-6}
Streaming services	10^{-7}
Video telephony, High speed computing	10^{-7}

In order to achieve such high levels of reliability we have to resort to the use of **Channel Coding**. The basic objective of channel coding is to increase the resistance of the digital communication system to channel noise. This is done by adding redundancies in the transmitted data stream in a controlled manner.

In channel coding, we map the incoming data sequence to a channel input sequence. This encoding procedure is done by the **Channel Encoder**. The encoded sequence is then transmitted over the noisy channel. The channel output sequence at the receiver is inverse mapped to an output data sequence. This is called the decoding procedure, and is carried out by the **Channel Decoder**. Both the encoder and the decoder are under the designer's control.

As already mentioned, the encoder introduces redundancy in a prescribed manner. The decoder exploits this redundancy so as to reconstruct the original source sequence as accurately as possible. Thus, channel coding makes it possible to carry out reliable communication over unreliable (noisy) channels. Channel coding is also referred to as **Error Control Coding**. It is interesting to note here that the source coder reduces redundancy to improve efficiency, whereas, the channel coder adds redundancy in a controlled manner to improve reliability.

Definition 2.6 An **Error Control Code** for a channel, represented by the channel transition probability matrix $p(y|x)$, consists of:

- (i) A message set $\{1, 2, \dots, M\}$.
- (ii) An encoding function, X^n , which maps *each* message to a unique **codeword**, i.e., $1 \rightarrow X^n(1)$, $2 \rightarrow X^n(2)$, ..., $M \rightarrow X^n(M)$. The set of codewords is called a **codebook**.
- (iii) A decoding function, $D \rightarrow \{1, 2, \dots, M\}$, which makes a guess based on a **decoding strategy** in order to map back the received vector to one of the possible messages.

We first look at a class of channel codes called *block codes*. In this class of codes, the incoming message sequence is first sub-divided into sequential blocks, each of length k bits. Thus the cardinality of the message set $M = 2^k$. Each k -bit long information block is mapped into an n -bit block by the channel coder, where $n > k$. This means that for every k bits of information, $(n - k)$ redundant bits are added. The ratio

$$r = \frac{k}{n} = \frac{\log M}{n} \quad (2.10)$$

is called the **Code Rate**. It represents the information bits per transmission. Code rate of any coding scheme is, naturally, less than unity. A small code rate implies that more and more bits per block are the redundant bits corresponding to a higher coding overhead. This may reduce the effect of noise, but will also reduce the communication rate as we will end up transmitting more of the redundant bits and fewer information bits. The question before us is whether there exists a coding scheme such that the probability that the message bit will be in error is arbitrarily small and yet the coding rate is not too small? The answer is yes, and was first provided by Shannon in his second theorem regarding the channel capacity. We will study this shortly.

Definition 2.7 A rate r is said to be **achievable** if there exists a coding scheme (n, k) such that the maximal probability of error tends to 0 as $n \rightarrow \infty$. The (n, k) code may also be expressed as a $(2^{nr}, n)$ code or a (M, n) code, where $M = 2^k = 2^{nr}$.

DID YOU KNOW ? Let us now introduce the concept of time in our discussion. We wish to look at questions like how many bits per second can we send over a given noisy channel with arbitrarily low bit error rates? Suppose the discrete memoryless source has the source alphabet X and entropy $H(X)$ bits per source symbol. Let the source generate a symbol every T_s seconds. Then the average information rate of the source is $\frac{H(X)}{T_s}$ bits per second. Let us assume that the channel can be used once every T_c seconds and the capacity of the channel is C bits per channel use. Then the channel capacity per unit time is $\frac{C}{T_c}$ bits per second. We now state Shannon's second theorem known as the **Noisy Channel Coding Theorem** or simply the **Channel Coding Theorem**.

Theorem 2.1 Noisy Channel Coding Theorem

- (i) Let a DMS with an alphabet X have entropy $H(X)$ and produce symbols every T_s seconds. Let a discrete memoryless channel have capacity C and be used once every T_c seconds. Then if

$$\frac{H(X)}{T_s} \leq \frac{C}{T_c} \quad (2.11)$$

exists a coding scheme for which the source output can be transmitted over the noisy channel and be reconstructed with an arbitrarily low probability of error.

- (ii) Conversely if

$$\frac{H(X)}{T_s} > \frac{C}{T_c} \quad (2.12)$$

It is not possible to transmit information over the channel and reconstruct it with an arbitrarily small probability of error.

The parameter $\frac{C}{T_c}$ is called the **Critical Rate**. Thus, all rates below capacity, C , are achievable.



The channel coding theorem is a very important result in information theory. The theorem specifies the channel capacity, C , as a fundamental limit on the rate at which reliable communication can be carried out over an unreliable (noisy) DMS channel. It should be noted that the channel coding theorem tells us about the existence of some codes that can achieve reliable communications in a noisy environment. Unfortunately, it does not give us the recipe to construct these codes. Therefore, channel coding is still an active area of research as the search for better and better codes is still going on. From the next chapter onwards we shall study some good channel codes.

Example 2.4 Consider a DMS source that emits equally likely binary symbols ($p = 0.5$) once every T_s seconds. This entropy for this binary source is

$$H(p) = -p \log_2 p - (1-p)\log_2(1-p) = 1 \text{ bit}$$

The information rate of this source is

$$\frac{H(X)}{T_s} = \frac{1}{T_s} \text{ bits/second}$$

Suppose we wish to transmit the source symbols over a noisy channel. The source sequence is applied to a channel coder with code rate r . This channel coder uses the channel once every T_c seconds to send the coded sequence. We want to have reliable communication (the probability of error as small as desired). From the channel coding theorem, if

$$\frac{1}{T_s} \leq \frac{C}{T_c} \quad (2.13)$$

we can make the probability of error as small as desired by a suitable choice of a channel coding scheme, and hence have reliable communication. We note that the ratio is equal to the code rate of the coder. Therefore,

$$r = \frac{T_c}{T_s} \quad (2.14)$$

Hence the condition for reliable communication can be rewritten as

$$r \leq C \quad (2.15)$$

Thus, for a BSC one can find a suitable channel coding scheme with a code rate, $r \leq C$, which will ensure reliable communication regardless of how noisy the channel is!



Interpretation

In general, all rates below capacity, C , are achievable. That is, for every rate $r \leq C$ there exists a $(2^n, n)$ code with arbitrarily low probability of error. When we say the probability is ‘as small as desired’ or ‘arbitrarily low’ we mean it literally! If we want the residual error rate (i.e., the error rate after error correction) to be 10^{-20} we can (in principle) achieve it as long as $r \leq C$. Shall we become more ambitious and desire that the residual error rate be 10^{-35} ?

No problem! We need to just ensure that $r \leq C$. Of course, we can state that at least one such code exists, but finding that code may not be a trivial job. As we shall see later, the level of noise in the channel will manifest itself by limiting the channel capacity, and hence the code rate.

Example 2.5 Consider a binary symmetric channel (BSC) with a transition probability $p = 10^{-2}$.

Such error rates are typical of wireless channels. We saw in Example 2.1 that for a BSC, the capacity is given by

$$C = 1 + p \log_2 p + (1 - p) \log_2 (1 - p)$$

By plugging in the value of $p = 10^{-2}$ we obtain the channel capacity $C = 0.919$. From the previous example we can conclude that there exists at least one coding scheme with the code rate $r \leq 0.919$ which will guarantee us a (non-zero) probability of error that is ‘as small as desired’. We mean it literally!

Example 2.6 Consider the **repetition code** in which each message bit is simply repeated n times, where n is an odd integer. For example, for $n = 3$, we have the mapping scheme

$$0 \rightarrow 000; 1 \rightarrow 111$$

Similarly, for $n = 5$ we have the mapping scheme

$$0 \rightarrow 00000; 1 \rightarrow 11111$$

Note that the code rate of the repetition code with blocklength n is

$$r = \frac{1}{n} \quad (2.16)$$

The decoding strategy is as follows: If in a block of n received bits the number of 0's exceeds the number of 1's, decide in favor of 0 and vice versa. This is also known as **Majority Decoding**. This also answers the question why should n be an odd integer for repetition codes.

Let $n = 2m + 1$, where m is a positive integer. This decoding strategy will make an error if more than m bits are in error, because in that case even if a 0 is encoded and sent, there will be more number of 1's in the received word. Let us assume that the *a priori* probabilities of 1 and 0 are equal. Then, the average probability of error is given by

$$P_e = \sum_{i=m+1}^n \binom{n}{i} p^i (1-p)^{n-i} \quad (2.17)$$

where p is the channel transition probability. The average probability of error for repetition codes for different code rates is given in Table 2.2.

Table 2.2 Average probability of error for repetition codes

Code Rate, r	Average Probability of Error, P_e
1	10^{-2}
1/3	3×10^{-4}
1/5	10^{-6}
1/7	4×10^{-7}
1/9	10^{-8}
1/11	5×10^{-10}



From Table 2.2, we see that as the code rate decreases, there is a steep fall in the average probability of error. The decrease in the P_e is much more rapid than the decrease in the code rate, r . However, for repetition codes, the code rate tends to zero if we want smaller and smaller P_e . Thus the repetition code exchanges code rate for message reliability. But the channel coding theorem states that the code rate need not tend to zero in order to obtain an arbitrarily low probability of error. The theorem merely requires the code rate r to be less than the channel capacity, C . So there must exist some code (other than the repetition code) with code rate $r \approx 0.9$ which can achieve arbitrarily low probability of error. Such a coding scheme will add just 1 parity bit to 9 information bits (or, maybe, add 10 extra bits to 90 information bits, or add 100 extra bits to 900 information bits, ...) and give us as small a P_e as desired (say, 10^{-20})! The hard part is finding such a code. Also, Shannon proved the noisy channel coding theorem using block codes with 'large' blocklengths, which may be impractical in real-life.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 3:

1. Consider the binary channel shown in Fig. 2.11. Let the *a priori* probabilities of sending the binary symbol be p_0 and p_1 , where $p_0 = p_1 = 0.5$. What is the maximum rate at which transmission can be carried out over this channel with arbitrarily low probability of error?
2. Consider a satellite hop channel shown in Fig. 2.12. The uplink can be modeled as a BSC with crossover probability p while the downlink can be modeled as a BSC with crossover probability q . Assuming the uplink channel and downlink channel are independent, draw the capacity region for this satellite hop channel. Hint: Choose the two orthogonal axes as r_1 and r_2 and mark a 2-D region where reliable communication is achievable.

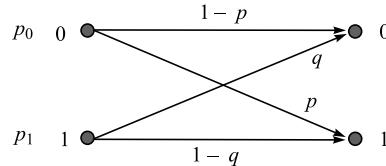


Fig. 2.11

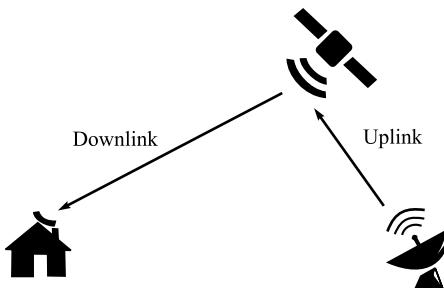


Fig. 2.12

If you have successfully solved the above problems,
you have mastered LO 3. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/585>



2.5 Information Capacity Theorem

So far we have studied limits on the maximum rate at which information can be sent over a channel reliably in terms of the channel capacity. In this section we will formulate the information capacity theorem for band-limited, power-limited Gaussian channels. An important and useful channel is the Gaussian channel defined below.

LO 4

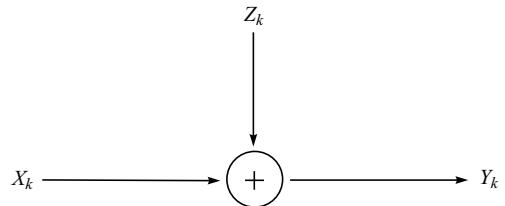


State and prove the Information Capacity Theorem and discuss its ramifications.

Definition 2.8 A Gaussian Channel

is a time discrete channel with output Y_k , at time k , which is the result of the sum of the input X_k and the Gaussian noise Z_k .

This noise is drawn from a Gaussian distribution with mean zero and variance σ^2 . Thus, $Y_k = X_k + Z_k$, where $Z_k \sim N(0, \sigma^2)$. The noise Z_k is independent of the input X_k . The Gaussian Channel is shown in Fig. 2.13.

**Fig. 2.13** Gaussian channel.

The Gaussian channel can be used to model a wide range of communication links. The noise in such channels could be because of various different causes. However, it is fair to assume that these random effects are independent and identically distributed and manifest themselves as creating electrical disturbances in the communication circuit. Because of the superposition theorem for electrical circuits, we can add up all these random effects. The net effect of the sum of these independent and identically distributed random effects is Gaussian from the central limit theorem.

Consider a zero mean, stationary random process $X(t)$ that is band limited to W hertz. Let $X_k, k = 1, 2, \dots, K$, denote the continuous random variables obtained by uniform sampling of the process $X(t)$ at the Nyquist rate of $2W$ samples per second. These symbols are transmitted over a noisy channel which is also band-limited to W Hertz. The channel output is corrupted by AWGN of zero mean and power spectral density (psd) $N_0/2$. Because of the channel, the noise is band limited to W Hertz. Let $Y_k, k = 1, 2, \dots, K$, denote the samples of the received signal. Therefore,

$$Y_k = X_k + N_k, k = 1, 2, \dots, K \quad (2.18)$$

where N_k is the noise sample with zero mean and variance $\sigma^2 = N_0 W$. It is assumed that $Y_k, k = 1, 2, \dots, K$, are statistically independent. Since the transmitter is usually power-limited, let us put a constraint on the average power in X_k :

$$E[X_k^2] = P, k = 1, 2, \dots, K \quad (2.19)$$

The information capacity of this band-limited, power-limited channel is the maximum of the mutual information between the channel input X_k and the channel output Y_k . The maximization has to be done over all distributions on the input X_k that satisfy the power constraint of equation (2.19). Thus, the information capacity of the channel (same as the channel capacity) is given by

$$C = \max_{f_{X_k}(x)} \{I(X; Y) | E[X_k^2] = P\} \quad (2.20)$$

where $f_{X_k}(x)$ is the probability density function of X_k .

Now, from Chapter 1, equation (1.32), we have,

$$I(X_k; Y_k) = h(Y_k) - h(Y_k | X_k) \quad (2.21)$$

Note that X_k and N_k are independent random variables. Therefore, the conditional differential entropy of Y_k given X_k is equal to the differential entropy of N_k . Intuitively, this is because given X_k the uncertainty arising in Y_k is purely due to N_k . That is,

$$h(Y_k | X_k) = h(N_k) \quad (2.22)$$

Hence we can write equation (2.21) as

$$I(X_k; Y_k) = h(Y_k) - h(N_k) \quad (2.23)$$

Since $h(N_k)$ is independent of X_k , maximizing $I(X_k; Y_k)$ translates to maximizing $h(Y_k)$. It can be shown that in order for $h(Y_k)$ to be maximum, Y_k has to be a Gaussian random variable. If we assume Y_k to be Gaussian, and N_k is Gaussian by definition, then X_k is also Gaussian. This is because the sum (or difference) of two Gaussian random variables is also Gaussian. Thus, in order to maximize the mutual information between the channel input X_k and the channel output Y_k , the transmitted signal should be Gaussian. Therefore, we can rewrite (2.20) as

$$C = I(X; Y) | E[X^2] = P \text{ and } X_k \text{ is Gaussian} \quad (2.24)$$

We know that if two independent Gaussian random variables are added, the variance of the resulting Gaussian random variable is the sum of the variances. Therefore, the variance of the received sample Y_k equals $P + N_0 W$. It can be shown that the differential entropy of a Gaussian random variable with variance σ^2 is $\frac{1}{2} \log_2(2\pi e \sigma^2)$. Therefore,

$$h(Y_k) = \frac{1}{2} \log_2 [2\pi e (P + N_0 W)] \quad (2.25)$$

and

$$h(N_k) = \frac{1}{2} \log_2 [2\pi e (N_0 W)] \quad (2.26)$$

Substituting these values of differential entropy for Y_k and N_k we get

$$C = \frac{1}{2} \log_2 \left(1 + \frac{P}{N_0 W} \right) \text{ bits per channel use} \quad (2.27)$$

We are transmitting $2W$ samples per second, i.e., the channel is being used $2W$ times in one second. Therefore, the information capacity can be expressed as

$$C = W \log_2 \left(1 + \frac{P}{N_0 W} \right) \text{ bits per second} \quad (2.28)$$

This basic formula for the capacity of the band-limited, AWGN waveform channel with a band-limited and average power-limited input was first derived by Shannon in 1948. It is known as the Shannon's third theorem, the **Information Capacity Theorem**. Since $N_0 W$ is the average noise power, $\text{SNR} = \frac{P}{N_0 W}$. Therefore, (2.28) can be succinctly expressed as $C = W \log_2(1 + \text{SNR})$.

Example 2.7 Consider telephone channel with bandwidth 4 kHz and $\text{SNR} \approx 25$ dB. Let us model it as a band-limited and power-limited Gaussian channel.

SNR in dB is calculated as $10 \log_{10}(\text{SNR})$. Hence, 25 dB $\text{SNR} = 10^{2.5} = 316.22$.

The capacity of this channel is $C = W \log_2(1 + \text{SNR}) = 4 \times 10^3 \log_2(1 + 316.22) = 33,238$ b/s.

If we increase the SNR by 1 dB (i.e., $\text{SNR} \approx 26$ dB), the capacity increases to 34,563 b/s.

So, a 1 dB increase in SNR bought us roughly an additional 1325 b/s.

The ITU V.90 modem, that operates on the telephone line, provides a data rate of 33,600 b/s. Thus, the ITU V.90 modem operates very close to the capacity of the channel! If the channel SNR is poor, a lower data rate is used by the modem.

Theorem 2.2 Information Capacity Theorem: The information capacity of a continuous channel of bandwidth W Hertz, perturbed by additive white Gaussian noise of power spectral density $N_0/2$ and limited in bandwidth to W , is given by

$$C = W \log_2 \left(1 + \frac{P}{N_0 W} \right) \text{ bits per second}$$

where P is the average transmitted power. This theorem is also called the **Channel Capacity Theorem**.



The **Information Capacity Theorem** is one of the important results in information theory. In a single formula one can see the trade off between the channel bandwidth, the average transmitted power and the noise power spectral density. Given the channel bandwidth and the SNR the channel capacity (bits/second) can be computed. This channel capacity is the fundamental limit on the rate of reliable communication for a power-limited, band-limited Gaussian channel. It should be kept in mind that in order to approach this limit, the transmitted signal must have statistical properties that are Gaussian in nature. Note that the terms channel capacity and information capacity have been used interchangeably.

Let us now derive the same result in a more intuitive manner. Suppose we have a coding scheme that results in an acceptably low probability of error. Let this coding scheme take k information bits and encode them into n bit long codewords. The total number of codewords is $M = 2^k$. Let the average power per bit be P . Thus the average power required to transmit an entire codeword is nP . Let these codewords be transmitted over a Gaussian channel with the noise variance equal to σ^2 . The received vector of n bits is also Gaussian with the mean equal to the transmitted codeword and the variance equal to $n\sigma^2$. Since the code is a good one (acceptable error rate), the vector lies inside a sphere of radius $\sqrt{n\sigma^2}$, centered on the transmitted codeword. This sphere itself is contained in a larger sphere of radius $\sqrt{n(P+\sigma^2)}$, where $n(P+\sigma^2)$ is the average power of the received vector.

This concept may be visualized as depicted in Fig. 2.14. There is a large sphere of radius $\sqrt{n(P+\sigma^2)}$ which contains M smaller spheres of radius $\sqrt{n\sigma^2}$. Here $M = 2^k$ is the total number of codewords. Each of these small spheres is centered on a codeword. These are called the **Decoding Spheres**. Any received word lying within a sphere is decoded as the codeword on which the sphere is centered. Suppose a codeword is transmitted over a noisy channel. Then there is a *high* probability that received vector will lie inside the correct decoding sphere (since it is a reasonably good code). The question arises: How many non-intersecting spheres can be packed inside the large sphere? The more number of spheres one can pack, the more efficient will be the code in terms of the code rate. This is known as the **Sphere Packing Problem**.

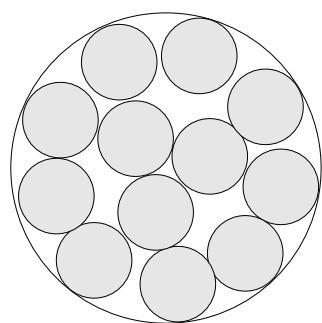


Fig. 2.14 Visualization of the sphere packing problem.

The volume of an n -dimensional sphere of radius r can be expressed as

$$V = A_n r^n \quad (2.29)$$

where A_n is a scaling factor. Therefore, the volume of the large sphere (sphere of all possible received vectors) can be written as:

$$V_{all} = A_n \left[n(P + \sigma^2) \right]^{n/2} \quad (2.30)$$

and the volume of the decoding sphere can be written as:

$$V_{ds} = A_n \left[n\sigma^2 \right]^{n/2} \quad (2.31)$$

The maximum number of non intersecting decoding spheres that can be packed inside the large sphere of all possible received vectors is

$$M = \frac{A_n \left[n(P + \sigma^2) \right]^{n/2}}{A_n \left[n\sigma^2 \right]^{n/2}} = \left(1 + \frac{P}{\sigma^2} \right) = 2^{(n/2)\log_2(1+P/\sigma^2)} \quad (2.32)$$

On taking logarithm (base 2) on both sides of the equation we get

$$\log_2 M = \frac{n}{2} \log_2 \left(1 + \frac{P}{\sigma^2} \right) \quad (2.33)$$

Observing that $k = \log_2 M$, we have

$$\frac{k}{n} = \frac{1}{2} \log_2 \left(1 + \frac{P}{\sigma^2} \right) \quad (2.34)$$

Note that each time we use the channel, we effectively transmit $\frac{k}{n}$ bits. Thus, the maximum number of bits that can be transmitted *per channel use*, with a low probability of error, is $\frac{1}{2} \log_2 \left(1 + \frac{P}{\sigma^2} \right)$, as seen previously in equation (2.27). Note that σ^2 represents the noise power and is equal to $N_0 W$ for AWGN with power spectral density $\frac{N_0}{2}$ and limited in bandwidth to W .

Example 2.8 Consider a communication system, as depicted in Fig. 2.15 with two Gaussian noise sources Z_1 and Z_2 that are independent, have zero mean and variances σ_1^2 and σ_2^2 respectively. Assume that X is constrained to have power $E[X^2] = P$.

Since the two noise sources Z_1 and Z_2 are independent, the effective noise can be modeled as Gaussian with zero mean and variance $\sigma_1^2 + \sigma_2^2$. Therefore, the capacity of this channel with two noise sources is simply

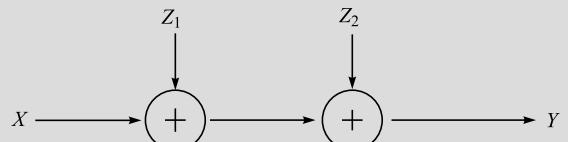


Fig. 2.15 Two noise sources.

$$C = \frac{1}{2} \log_2 \left(1 + \frac{P}{\sigma_1^2 + \sigma_2^2} \right) \text{ bits per channel use.}$$

Now, suppose Z_2 is known to the receiver. In this case, the noise Z_2 can be subtracted at the receiver, thereby yielding the capacity

$$C = \frac{1}{2} \log_2 \left(1 + \frac{P}{\sigma_1^2} \right) \text{ bits per channel use.}$$

Example 2.9 Consider two parallel Gaussian

channels as shown in Fig 2.16. The input X is transmitted over these Gaussian channels. X is constrained to have power $E[X^2] = P$. The noises Z_1 and Z_2 are independent and have variances σ_1^2 and σ_2^2 .

At the receiver we obtain Y_1 and Y_2 . As a first step, let us combine the outputs to produce

$$Y = Y_1 + Y_2 = X + X + (Z_1 + Z_2) = 2X + Z,$$

where $Z = Z_1 + Z_2$. This is a Gaussian channel with received power $= E[(2X)^2] = 4P$.

Since Z_1 and Z_2 are independent, the total noise power is the sum of the variances

$$\sigma^2 = \sigma_1^2 + \sigma_2^2.$$

The capacity of the X to Y channel is the Shannon capacity given by

$$C = \frac{1}{2} \log_2 \left(1 + \frac{4P}{\sigma_1^2 + \sigma_2^2} \right) \text{ bits per channel use}$$



Can we do better? As the next step, let us use the following strategy for combining the received signals

$$Y = \alpha Y_1 + (1 - \alpha) Y_2$$

where $0 \leq \alpha \leq 1$. Thus we have

$$Y = \alpha(X + Z_1) + (1 - \alpha)(X + Z_2) = X + Z,$$

where $Z = \alpha Z_1 + (1 - \alpha) Z_2$. This is a Gaussian channel with received power $= E[(X)^2] = P$. The total noise power is given by

$$\sigma^2 = \alpha^2 \sigma_1^2 + (1 - \alpha)^2 \sigma_2^2$$

In order to maximize capacity, we must minimize the total noise power. Setting $\frac{d\sigma^2}{d\alpha} = 0$ we obtain

$$\alpha_{opt} = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2}$$

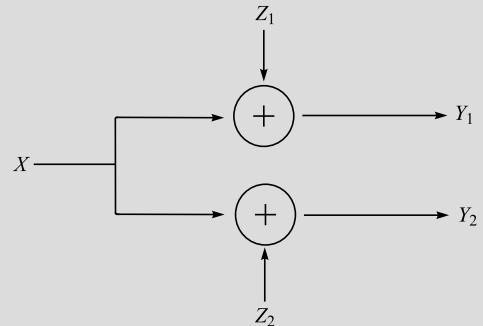


Fig. 2.16 Parallel gaussian channels.

and the minimum total noise power $\sigma_{min}^2 = \frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2}$. Using this, we obtain the maximum capacity as

$$C' = \frac{1}{2} \log_2 \left(1 + \frac{P}{\sigma_1^2} + \frac{P}{\sigma_2^2} \right) \text{ bits per channel use}$$

One can show that $C' \geq C$, with the condition for equality, $\sigma_1^2 = \sigma_2^2$. This implies that if the two channels are not equally good ($\sigma_1^2 \neq \sigma_2^2$), one must use weighted combining in order to maximize capacity.

In the previous example, equal power is transmitted over the two independent channels. In the case when the channels are not equally good (or bad!), we do optimal combining. To do so, we need the **Channel State Information** (CSI). Can we use the CSI at the transmitter to increase the capacity? We explore that exciting option in the next section.

2.6 Parallel Gaussian Channels

LO 4

Consider k independent parallel Gaussian channels, as depicted in Fig. 2.17. Let us assume that there exists a constraint on the total transmit power. The aim is to distribute the total available power among the k channels so as to maximize the capacity.

The output of channel j is given by

$$Y_j = X_j + Z_j \quad (2.35)$$

with $Z_i \sim N(0, \sigma_j^2)$. The noise in each channel is assumed to be independent. The total power constraint can be expressed as

$$E \sum_{j=1}^k X_j^2 \leq P \quad (2.36)$$

The information capacity of these k independent parallel Gaussian channels can be written as

$$C = \max_{f(x_1, x_2, \dots, x_k), E[X_i^2] \leq P} I(X_1, X_2, \dots, X_k; Y_1, Y_2, \dots, Y_k) \quad (2.37)$$

We can simplify the right hand side as follows:

$$\begin{aligned} & I(X_1, X_2, \dots, X_k; Y_1, Y_2, \dots, Y_k) \\ &= h(Y_1, Y_2, \dots, Y_k) - h(Y_1, Y_2, \dots, Y_k | X_1, X_2, \dots, X_k) \end{aligned} \quad (2.38)$$

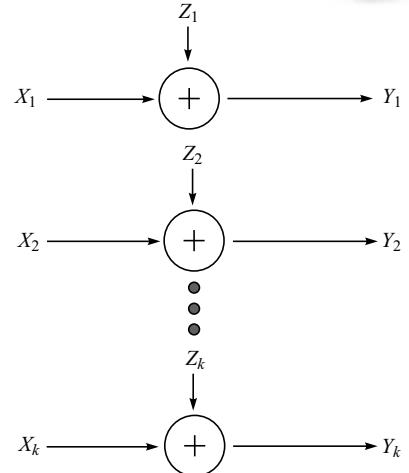


Fig. 2.17 Parallel independent gaussian channels.

Since the uncertainty in the outputs $\{Y_1, Y_2, \dots, Y_k\}$, given the inputs $\{X_1, X_2, \dots, X_k\}$, depends only on the noise $\{Z_1, Z_2, \dots, Z_k\}$, we can write

$$\begin{aligned} & I(X_1, X_2, \dots, X_k; Y_1, Y_2, \dots, Y_k) \\ &= h(Y_1, Y_2, \dots, Y_k) - h(Z_1, Z_2, \dots, Z_k | X_1, X_2, \dots, X_k) \\ &= h(Y_1, Y_2, \dots, Y_k) - h(Z_1, Z_2, \dots, Z_k) = h(Y_1, Y_2, \dots, Y_k) - \sum_{i=1}^k h(Z_i) \end{aligned} \quad (2.39)$$

The last step follows from the assumption that the noise is independent of the input in the channel, and the noise in each channel is independent. Using the fact that $h(Y_1, Y_2, \dots, Y_k) \leq \sum_{i=1}^k h(Y_i)$, we have

$$\begin{aligned} & I(X_1, X_2, \dots, X_k; Y_1, Y_2, \dots, Y_k) \\ &\leq \sum_{i=1}^k h(Y_i) - \sum_{i=1}^k h(Z_i) \\ &\leq \sum_{i=1}^k \frac{1}{2} \log \left(1 + \frac{P_i}{\sigma_i^2} \right) \end{aligned} \quad (2.40)$$

where $P_i = E[X_i^2]$ and $\sum_{i=1}^k P_i = P$. For equality in the last step, $X_i \sim N(0, P_i)$ for $i = 1, 2, \dots, k$. Maximizing the power allocation in each channel in order to maximize the capacity, we obtain

$$\begin{aligned} & P_i = (\pi - \sigma_i^2)^+ \text{ such that} \\ & \sum_{i=1}^k (\pi - \sigma_i^2)^+ = P \end{aligned} \quad (2.41)$$

Here, $(x)^+$ is defined as

$$(x)^+ = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \quad (2.42)$$


From (2.41), the optimal power allocation strategy can be visualized as follows. Channel j can be assumed to be a solid block of height σ_j^2 . All blocks (channels) are placed inside a container, as shown in Fig. 2.18(a). We now pour water (power) into the container such that the water level in the container rises to π , as depicted in Fig. 2.18(b). The water level above each block (channel) indicates how much power is to be allocated to that channel. For example, channel j with noise power σ_j^2 gets $(\pi - \sigma_j^2)$ power allocated to it. Those blocks (channels) that are above the water level π do not get any power allocated to them. Because of this physical analogy, this algorithm is also known as the **Water Filling or Water Pouring Algorithm**.

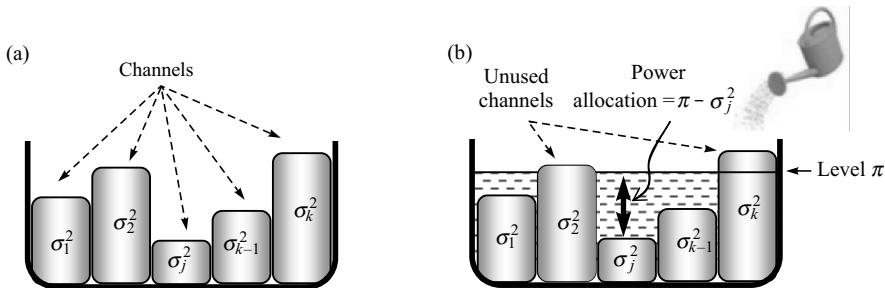


Fig. 2.18 (a) The channel noise power as solid blocks. (b) After water filling, only three channels get power allocations.

2.7 The Shannon Limit

LO 4

Consider a Gaussian channel that is limited both in power and bandwidth. We wish to explore the limits of a communication system under these constraints. Let us define an ideal system which can transmit data at a bit rate R_b which is equal to the capacity, C , of the channel, i.e., $R_b = C$. Suppose the energy per bit is E_b . Then the average transmitted power is

$$P = E_b R_b = E_b C \quad (2.43)$$

Therefore, the channel capacity theorem for this ideal system can be written as:



$$\frac{C}{W} = \log_2 \left(1 + \frac{E_b}{N_0 W} \right) \quad (2.44)$$

This equation can be re-written in the following form:

$$\frac{E_b}{N_0} = \frac{2^{C/W} - 1}{C/W} \quad (2.45)$$

The plot of the bandwidth efficiency $\frac{R_b}{W}$ versus $\frac{E_b}{N_0 W}$ is called the **Bandwidth Efficiency Diagram**, and is given in Fig. 2.19. The ideal system is represented by the line $R_b = C$.

The following conclusions can be drawn from the bandwidth efficiency diagram.

- (i) For infinite bandwidth, the ratio $\frac{E_b}{N_0}$ tends to the limiting value

$$\left. \frac{E_b}{N_0} \right|_{W \rightarrow \infty} = \ln 2 = 0.693 = -1.6 \text{ dB} \quad (2.46)$$

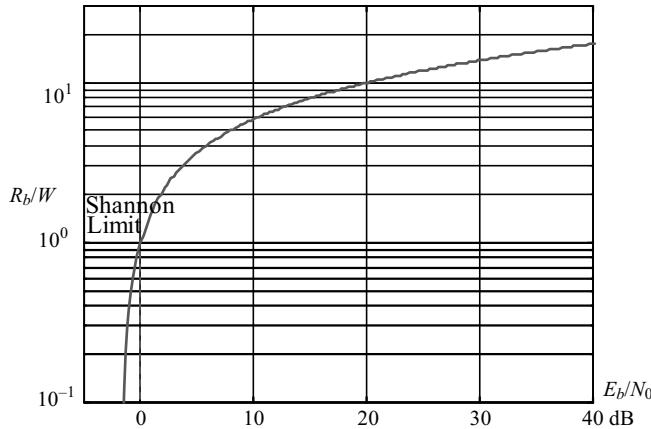


Fig. 2.19 The bandwidth efficiency diagram.

This value is called the **Shannon Limit**. It is interesting to note that the Shannon limit is a fraction. This implies that for very large bandwidths, reliable communication is possible even for the case when the signal power is *less than* the noise power! The channel capacity corresponding to this limiting value is

$$C|_{W \rightarrow \infty} = \frac{P}{N_0} \log_2 e \quad (2.47)$$

Thus, at infinite bandwidth, the capacity of the channel is determined by the SNR.

- (ii) The curve for the critical rate $R_b = C$ is known as the **Capacity Boundary**. For the case $R_b > C$, reliable communication is not guaranteed. However, for $R_b < C$, there exist some coding scheme which can provide an arbitrarily low probability of error.
- (iii) The bandwidth efficiency diagram shows the trade-offs between the quantities $\frac{R_b}{W}$, $\frac{E_b}{N_0}$ and the probability of error, P_e . Note that for designing any communication system the basic design parameters are the bandwidth available, the SNR and the bit error rate (BER). The BER is determined by the application and the quality of service (QoS) desired. The bandwidth and the power can be traded one for the other to provide the desired BER.
- (iv) Any point on the bandwidth efficiency diagram corresponds to an operating point corresponding to a set of values of SNR, Bandwidth efficiency and BER.



Intuition The information capacity theorem predicts the maximum amount of information that can be transmitted through a given bandwidth for a given SNR. We see from Fig. 2.19 that acceptable capacity can be achieved even for low SNRs, provided adequate bandwidth is available. We also note that the capacity is a linear function of bandwidth and a logarithmic function of SNR. Thus the optimum usage of a given bandwidth is obtained when the signals are noise-like (i.e., of large bandwidth) and a minimal SNR is maintained at the receiver. This principle lies in the heart of any spread spectrum communication system, like **Code Division Multiple Access** (CDMA). In the light of this theorem one can justify why all the Third Generation (3G) mobile systems are CDMA based.

2.8 Channel Capacity for MIMO Systems

LO4

Let us model the MIMO channel described by (2.4) using the matrix \mathbf{H} of dimension $M_R \times M_T$. Assuming the average transmit symbol energy is E_s , the sampled signal model can be expressed as

$$\mathbf{y}[k] = \sqrt{\frac{E_s}{M_T}} \mathbf{H}\mathbf{s}[k] + \mathbf{n}[k] \quad (2.48)$$

where $\mathbf{y}[k]$ is the received signal vector with dimension $M_R \times 1$, $\mathbf{s}[k]$ is the transmit signal vector with dimension $M_T \times 1$ and $\mathbf{n}[k]$ is the $M_R \times 1$ spatio-temporal zero mean, complex Gaussian white noise vector with variance N_0 in each dimension. This signal model is true for **Frequency Flat Fading** channels. We can drop the time index k for clarity and write the above equation as

$$\mathbf{y} = \sqrt{\frac{E_s}{M_T}} \mathbf{H}\mathbf{s} + \mathbf{n} \quad (2.49)$$

The covariance matrix of \mathbf{s} is given by

$$\mathbf{R}_{ss} = E\{\mathbf{s}\mathbf{s}^H\} \quad (2.50)$$

The superscript ‘ H ’ denotes the **Hermitian** operation. We make the assumptions that the channel, \mathbf{H} , is deterministic and known to the receiver. The **Channel State Information** (CSI) can be obtained at the receiver using a pilot or training signals. The capacity of the MIMO channel is given by

$$C = \max_{\text{Tr}(\mathbf{R}_{ss})=M_T} W \log_2 \det \left(\mathbf{I}_{M_R} + \frac{E_s}{M_T N_0} \mathbf{H} \mathbf{R}_{ss} \mathbf{H}^H \right) \text{ bits per second} \quad (2.51)$$

where W is the bandwidth and \mathbf{I}_{M_R} denotes the identity matrix of size M_R . The condition $\text{Tr}(\mathbf{R}_{ss}) = M_T$ constrains the total average energy transmitted over a symbol period.

If the channel is unknown to the transmitter, the vector \mathbf{s} may be chosen such that $\mathbf{R}_{ss} = \mathbf{I}_{M_T}$. This simply means that the signals at the transmit antennas are independent and of equal power. The capacity of the MIMO channel in this case is given by



$$C = W \sum_{i=1}^r \log_2 \left(1 + \frac{E_s}{M_T N_0} \lambda_i \right) \text{ bits per second} \quad (2.52)$$

where r is the rank of the channel and λ_i ($i = 1, 2, \dots, r$) are the positive eigenvalues of $\mathbf{H}\mathbf{H}^H$. It is interesting to observe that the capacity of the MIMO channel unknown to the transmitter comes out as the sum of r SISO channels, each having power gain λ_i ($i = 1, 2, \dots, r$) and equal transmit power, $\frac{E_s}{M_T}$. It can be interpreted as

follows. The use of multiple transmit and receive antennas have opened multiple parallel data pipes between the transmitter and receiver. The number of these scalar data pipes depends on the rank of \mathbf{H} .

Next, consider a full-rank MIMO channel with $M_T = M_R = M$, so that $r = M$. The maximum capacity is achieved when \mathbf{H} is an orthogonal matrix, i.e., $\mathbf{H}\mathbf{H}^H = \mathbf{H}^H\mathbf{H}$. The capacity of this MIMO channel is given by

$$C = WM \log_2 \left(1 + \frac{E_s}{N_0} \right) \text{ bits per second} \quad (2.53)$$

The capacity of an orthogonal MIMO channel is simply M times the scalar channel capacity.

 If the channel is known to the transmitter, the different scalar data pipes may be accessed individually through processing at the transmitter and receiver. The basic idea is to allocate variable energy across the different data pipes in order to maximize the mutual information. The optimal energy can be found iteratively using the **water pouring algorithm**. The capacity of a MIMO channel when the channel is known to the transmitter is necessarily greater than or equal to the capacity when the channel is unknown to the transmitter.

INDUSTRIAL RELEVANCE



Long Term Evolution (LTE) Advanced has the provision for using 8×8 MIMO in the downlink and 4×4 MIMO in the uplink. In LTE MIMO, precoding is used to map the modulation symbols onto the different antennas. The specific type of precoding depends on the multi-antenna technique used.

2.9 Capacity Region for Multiple Access Channels

LO 4

Multiple access channels are commonly found in the domain of cellular communications where several mobile phone users try to access the same base station over a common channel. Similarly, a satellite receiver with several independent ground weather stations constitutes a multiple access channel. We note that, in addition to noise, there is interference generated by the multiple users (one user's signal is someone else's interference!). In a classroom scenario, several students asking questions to the instructor at the same time also tantamount to a multiple access channel. Clearly, if two students ask two different questions simultaneously, the instructor will have a difficult time understanding either of them unless the students use some form of clever coding.

Definition 2.9 A discrete memoryless **Multiple Access Channel** with two senders and one receiver can be characterized by three alphabets, X_1 , X_2 and Y and a corresponding probability transition matrix $p(y|x_1, x_2)$.

Suppose, the rates of the two transmitters are r_1 and r_2 respectively. Let the two transmitters accessing the common channel use two encoding functions, E_1 and E_2 in order to encode message sets $M_1 = \{1, 2, \dots, 2^{m_1}\}$ and $M_2 = \{1, 2, \dots, 2^{m_2}\}$ into X_1 and X_2 , i.e.,

$$E_1: M_1 \rightarrow X_1 \text{ and } E_2: M_2 \rightarrow X_2 \quad (2.54)$$

where n is the length of the encoded sequence X_i , $i = 1, 2$. The receiver needs to have a decoding function D such that

$$D: Y \rightarrow X_1 \times X_2 \quad (2.55)$$

Definition 2.10 For a multiple access channel, a rate pair (r_1, r_2) is **achievable** if there exists codes $((2^{m_1}, 2^{m_2}), n)$ such that the average probability of error $P_e \rightarrow 0$.

Definition 2.11 For a multiple access channel, the **Capacity Region** is the closure of the set of achievable rate pairs (r_1, r_2) .

Example 2.10 Consider a multiple access channel with two independent senders and one receiver. Suppose sender one has a binary symmetric channel with crossover probability p_1 , and sender two has a binary symmetric channel with crossover probability p_2 . If the two channels are indeed independent, then they experience no interference from each other (they could be using orthogonal frequencies). Thus, sender one can send information at rate $C_1 = 1 - H(p_1)$ and similarly, sender two can send information at rate $C_2 = 1 - H(p_2)$. The capacity region is shown in Fig. 2.20.

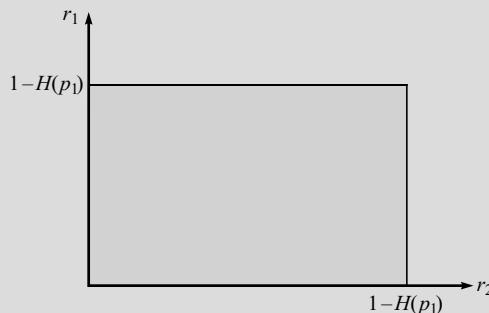


Fig. 2.20 The capacity region for independent binary symmetric channels.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 4:

1. A telephone channel has a bandwidth of 3000 Hz and the SNR = 20 dB. S
 (i) Determine the channel capacity.
 (ii) If the SNR is increased to 25 dB, determine the increased capacity.
2. Suppose a TV displays 30 frames/second. There are approximately 2×10^5 pixels per frame, each pixel requiring 16 bits for color display. Assuming an SNR of 25 dB calculate the bandwidth required to support the transmission of the TV video signal. M
3. (i) Prove that for a finite variance σ^2 , the Gaussian random variable has the largest differential entropy attainable by any random variable.
 (ii) Show that this entropy is given by $\frac{1}{2} \log_2(2\pi e\sigma^2)$. M
4. Consider a channel consisting of two parallel AWGN channels with inputs X_1 , X_2 and outputs D

$$Y_1 = X_1 + Z_1$$

$$Y_2 = X_2 + Z_2$$

The noises Z_1 and Z_2 are independent and have variances σ_1^2 and σ_2^2 with $\sigma_1^2 < \sigma_2^2$. However, we are constrained to use the same symbol on both channels, i.e. $X_1 = X_2 = X$, where X is constrained to have power $E[X^2] = P$. Suppose at the receiver, we combine the outputs to produce $Y = Y_1 + Y_2$. What is the capacity C_1 of channel with input X and output Y ?

5. Consider two parallel channels with independent Gaussian noise Z_1 and Z_2 with variances $\sigma_1^2 = 1$ and $\sigma_2^2 = 2$. The signal at the receiver is given by D

$$Y_1 = X_1 + Z_1$$

$$Y_2 = X_2 + Z_2$$

The transmitter is subject to the power constraint $E[X_1^2 + X_2^2] \leq P$.

Find and sketch the capacity $C(P)$ of this channel as a function of P . What happens when P becomes very small?

If you have successfully solved the above problems,
you have mastered LO 4. Congratulations!



: The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/586>



2.10 Random Selection of Codes

LO 5



Discuss random selection of codes and define the cutoff rate.

Consider a set of M coded signal waveforms constructed from a set of n -dimensional binary codewords. Let us represent these codewords as follows:

$$\mathbf{C}_i = [c_{i1} c_{i2} \dots c_{in}], i = 1, 2, \dots, M \quad (2.56)$$

Since we are considering binary codes, c_{ij} is either a 0 or a 1. Let each bit of the codeword be mapped on to a BPSK waveform so that the codeword may be represented as

$$s_i(t) = \sum_{j=1}^n s_{ij} p_j(t), i = 1, 2, \dots, M \quad (2.57)$$

where

$$s_{ij} = \begin{cases} \sqrt{E} & \text{for } c_{ij} = 1 \\ -\sqrt{E} & \text{for } c_{ij} = 0 \end{cases} \quad (2.58)$$

and \sqrt{E} is the energy per code bit. The waveform $s_i(t)$ can then be represented as the n -dimensional vector

$$\mathbf{s}_i = [s_{i1} \ s_{i2} \ \dots \ s_{in}], i = 1, 2, \dots, M \quad (2.59)$$

We observe that this corresponds to a hypercube in the n -dimensional space. Let us now encode k bits of information into an n bit long codeword, and map this codeword to one of the M waveforms. Note that there are a total of 2^k possible waveforms corresponding to the $M = 2^k$ different codewords.

Let the information rate into the encoder be R bits/sec. The encoder takes in k bits at a time and maps the k -bit block to one of the M waveforms. Thus $k = RT$ and $M = 2^k$ signals are required. Let us define a parameter D as follows:

$$D = \frac{n}{T} \text{ dimensions/sec} \quad (2.60)$$

$n = DT$ is the **dimensionality** of the space. The hypercube mentioned above has $2^n = 2^{DT}$ vertices. Of these, we must choose $M = 2^{RT}$ to transmit the information. Under the constraint $D > R$, the fraction of vertices that can be used as signal points is

$$F = \frac{2^k}{2^n} = \frac{2^{RT}}{2^{DT}} = 2^{-(D-R)T} \quad (2.61)$$

For $D > R$, $F \rightarrow 0$ as $T \rightarrow \infty$. Since, $n = DT$, it implies that $F \rightarrow 0$ as $n \rightarrow \infty$. Designing a good coding scheme translates to choosing M vertices out of the 2^n vertices of the hypercube in such a manner that the probability of error tends to zero as we increase n . We saw that the fraction F tends to zero as we choose larger and larger n . This implies that it is possible to increase the minimum distance between these M signal points as $n \rightarrow \infty$. Increasing the minimum distance between the signal points would give us the probability of error, $P_e \rightarrow 0$.



There are $(2^n)^M$ distinct ways of choosing M out of the total 2^n vertices. Each of these choices corresponds to a coding scheme. For each set of M waveforms, it is possible to design a communication system consisting of a modulator and a demodulator. Thus there are $2^n M$ communication systems, one for each choice of the M coded waveforms. Each of these communication systems is characterized by its probability of error. Of course, many of these communication systems will perform poorly in terms of the probability of error.

Let us pick one of the codes at random from the possible 2^{nM} sets of codes. The random selection of this m^{th} codeword occurs with the probability

$$P(\{\mathbf{s}_i\}_m) = \frac{1}{2^{nM}} \quad (2.62)$$

Let the corresponding probability of error for this choice of code be $P_e(\{\mathbf{s}_i\}_m)$. Then the average probability of error over the ensemble of codes is

$$\bar{P}_e = \sum_{m=1}^{2^{nM}} P_e(\{\mathbf{s}_i\}_m) P(\{\mathbf{s}_i\}_m) = \frac{1}{2^{nM}} \sum_{m=1}^{2^{nM}} P_e(\{\mathbf{s}_i\}_m) \quad (2.63)$$

We will next try to upper bound this average probability of error. If we have an upper bound on \bar{P}_e , then we can conclude that there exists at least one code for which this upper bound will also hold. Furthermore, if $\bar{P}_e \rightarrow 0$ as $n \rightarrow \infty$, we can surmise that $P_e(\{\mathbf{s}_i\}_m) \rightarrow 0$ as $n \rightarrow \infty$.

Consider the transmission of a k -bit message $\mathbf{X}_k = [x_1 x_2 \dots x_k]$ where x_j is binary for $j = 1, 2, \dots, k$. The conditional probability of error averaged over all possible codes is

$$\overline{P_e(\mathbf{X}_k)} = \sum_{\text{all codes}} P_e(\mathbf{X}_k, \{\mathbf{s}_i\}_m) P(\{\mathbf{s}_i\}_m) \quad (2.64)$$

where $P_e(\mathbf{X}_k, \{\mathbf{s}_i\}_m)$ is the conditional probability of error for a given k -bit message $\mathbf{X}_k = [x_1 x_2 \dots x_k]$, which is transmitted using the code $\{\mathbf{s}_i\}_m$. For the m^{th} code,

$$P_e(\mathbf{X}_k, \{\mathbf{s}_i\}_m) \leq \sum_{\substack{l=1 \\ l \neq k}}^M P_{2m}(\mathbf{s}_l, \mathbf{s}_k) \quad (2.65)$$

where, $P_{2m}(\mathbf{s}_l, \mathbf{s}_k)$ is the probability of error for the binary communication system using signal vectors \mathbf{s}_l and \mathbf{s}_k to transmit one of two equally likely k -bit messages. Hence,

$$\overline{P_e(\mathbf{X}_k)} \leq \sum_{\text{all codes}} P_e(\{\mathbf{s}_i\}_m) \sum_{\substack{l=1 \\ l \neq k}}^M P_{2m}(\mathbf{s}_l, \mathbf{s}_k) \quad (2.66)$$

On changing the order of summation we obtain

$$\overline{P_e(\mathbf{X}_k)} \leq \sum_{l=1}^M \left[\sum_{\substack{\text{all codes} \\ l \neq k}} P_e(\{\mathbf{s}_i\}_m) P_{2m}(\mathbf{s}_l, \mathbf{s}_k) \right] \leq \sum_{l=1}^M \overline{P_2(\mathbf{s}_l, \mathbf{s}_k)} \quad (2.67)$$

where $\overline{P_2(\mathbf{s}_l, \mathbf{s}_k)}$ represents the ensemble average of $P_{2m}(\mathbf{s}_l, \mathbf{s}_k)$ over the 2^{nM} codes. For additive white Gaussian noise channel,

$$P_{2m}(\mathbf{s}_l, \mathbf{s}_k) = Q\left(\sqrt{\frac{d_{lk}^2}{2N_0}}\right) \quad (2.68)$$

where $d_{lk}^2 = |\mathbf{s}_l - \mathbf{s}_k|^2$ and $Q(x) = \frac{1}{2\pi} \int_x^\infty e^{-z^2/2} dz$. If \mathbf{s}_l and \mathbf{s}_k differ in d coordinates,

$$d_{lk}^2 = |\mathbf{s}_l - \mathbf{s}_k|^2 = \sum_{j=1}^n (s_{lj} - s_{kj})^2 = d(2\sqrt{E})^2 = 4dE \quad (2.69)$$

Therefore,

$$P_{2m}(\mathbf{s}_l, \mathbf{s}_k) = Q\left(\sqrt{\frac{2dE}{N_0}}\right) \quad (2.70)$$

Under the assumption that all codes are equally probable, it is equally likely that the vector s_l is any of the 2^n vertices of the hypercube. Further, s_l and s_k are statistically independent. Hence, the probability that s_l and s_k differ in exactly d places is

$$P(d) = \left(\frac{1}{2}\right)^n \binom{n}{d} \quad (2.71)$$

The expected value of $P_{2m}(s_l, s_k)$ over the ensemble of codes is then given by

$$P_2(s_l, s_k) = \sum_{d=0}^n P(d) Q\left(\sqrt{\frac{2dE}{N_0}}\right) = \left(\frac{1}{2^n}\right) \sum_{d=0}^n \binom{n}{d} Q\left(\sqrt{\frac{2dE}{N_0}}\right) \quad (2.72)$$

Using the following upper bound

$$Q\left(\sqrt{\frac{2dE}{N_0}}\right) < e^{-\frac{dE}{N_0}} \quad (2.73)$$

we obtain

$$\overline{P_2(s_l, s_k)} \leq \left(\frac{1}{2^n}\right) \sum_{d=0}^n \binom{n}{d} e^{-\frac{dE}{N_0}} \leq \left[\frac{1}{2} \left(1 + e^{-\frac{E}{N_0}} \right) \right]^n \quad (2.74)$$

From (2.67) and (2.74) we obtain

$$\overline{P_e(X_k)} \leq \sum_{\substack{l=1 \\ l \neq k}}^M \overline{P_2(s_l, s_k)} = (M-1) \left[\frac{1}{2} \left(1 + e^{-\frac{E}{N_0}} \right) \right]^n < M \left[\frac{1}{2} \left(1 + e^{-\frac{E}{N_0}} \right) \right]^n \quad (2.75)$$

Recall that we need an upper bound on \bar{P}_e , the average error probability. To obtain \bar{P}_e we average $\overline{P_e(X_k)}$ over all possible k -bit information sequences. Thus,

$$\bar{P}_e = \sum_k \overline{P_e(X_k)} P(X_k) < M \left[\frac{1}{2} \left(1 + e^{-\frac{E}{N_0}} \right) \right]^n \sum_k P(X_k) < M \left[\frac{1}{2} \left(1 + e^{-\frac{E}{N_0}} \right) \right]^n \quad (2.76)$$

We now define a new parameter as follows.

Definition 2.12 The **Cutoff Rate** R_0 is defined as



$$R_0 = \log_2 \frac{2}{1 + e^{-\frac{E}{N_0}}} = 1 - \log_2 \left(1 + e^{-\frac{E}{N_0}} \right) \quad (2.77)$$

The cutoff rate has the units of bits/dimension. Observe that $0 \leq R_0 \leq 1$. The plot of R_0 with respect to SNR per dimension is given in Fig. 2.21.

The equation (2.76) can now be written succinctly as

$$\bar{P}_e < M 2^{-nR_0} = 2^{-RT} 2^{-nR_0} \quad (2.78)$$

Substituting $n = DT$ we obtain

$$\bar{P}_e < 2^{-T(DR_0 - R)} \quad (2.79)$$

If we substitute $T = n/D$, we obtain

$$\bar{P}_e < 2^{-n(R_0 - R/D)} \quad (2.80)$$

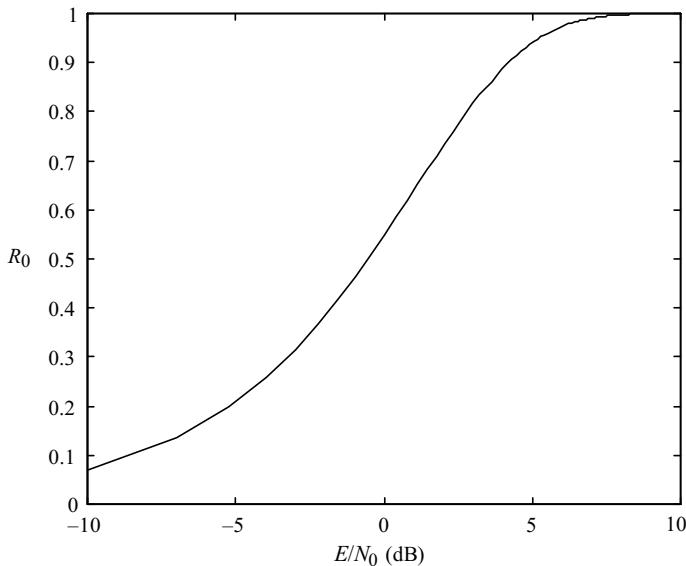


Fig. 2.21 Cutoff rate, R_0 , versus the SNR (in dB) per dimension.

Observe that

$$\frac{R}{D} = \frac{R}{n/T} = \frac{RT}{n} = \frac{k}{n} = R_c \quad (2.81)$$

Here R_c represents the code rate. Hence the average error probability can be written in the following instructive form

$$\bar{P}_e < 2^{-n(R_0 - R_c)} \quad (2.82)$$



From the above equation, we can conclude the following:

- Interpretation**
- (i) For $R_c < R_0$ the average probability of error $\bar{P}_e \rightarrow 0$ as $n \rightarrow \infty$. Since by choosing large values of n , \bar{P}_e can be made arbitrarily small, there exist good codes in the ensemble (which have the probability of error less than \bar{P}_e).
 - (ii) We observe the fact that \bar{P}_e is the ensemble average. Therefore, if a code is selected at random, the probability that its error $P_e > \alpha \bar{P}_e$ is less than $1/\alpha$. This implies that there are no more than 10% of the codes that have an error probability that exceeds $10 \bar{P}_e$. Thus, there are many good codes.

- (iii) The codes whose probability of error exceed \bar{P}_e are not always bad codes. The probability of error of these codes may be reduced by increasing the dimensionality, n .

For binary coded signals, the cutoff rate, R_0 , saturates at 1 bit/dimension for large values of $\frac{E}{N_0}$, say $\frac{E}{N_0} > 10$. Thus, to achieve lower probabilities of error one must reduce the code rate, R_c . Alternately, very

large block lengths have to be used. This is not an efficient approach. So, binary codes become inefficient at high SNRs. For high SNR scenarios, non-binary coded signal sets should be used to achieve an increase in the number of bits per dimension. Multiple-amplitude coded signal sets can be easily constructed from non-binary codes by mapping each code element into one of the possible amplitude levels (e.g. PAM). For random codes using M -ary multi-amplitude signals, it was shown by Shannon (in 1959) that

$$R_0^* = \frac{1}{2} \left[1 + \frac{E}{N_0} - \sqrt{1 + \left(\frac{E}{N_0} \right)^2} \right] \log_2 e + \frac{1}{2} \log_2 \left[\frac{1}{2} \left(1 + \sqrt{1 + \left(\frac{E}{N_0} \right)^2} \right) \right] \quad (2.83)$$

Let us now relate the cutoff rate R_0^* to the capacity of the AWGN channel, which is given by

$$C = W \log_2 \left(1 + \frac{P}{N_0 W} \right) \text{ bits per second} \quad (2.84)$$

where P is the average power and W is the channel bandwidth. The energy per code bit is equal to

$$E = \frac{PT}{n} \quad (2.85)$$

Recall that from the sampling theorem, a signal of bandwidth W may be represented by samples taken at a rate $2W$. Thus, in the time interval of length T there are $n = 2WT$ samples. Therefore we may write

$$D = \frac{n}{T} = 2W. \text{ Hence,}$$

$$P = \frac{nE}{T} = DE \quad (2.86)$$

Define normalized capacity, $C_n = \frac{C}{2W} = \frac{C}{D}$ and substitute for W and P in (2.84) to obtain

$$\begin{aligned} C_n &= \left(\frac{1}{2} \right) \log_2 \left(1 + 2 \frac{E}{N_0} \right) \\ &= \left(\frac{1}{2} \right) \log_2 (1 + 2R_c \gamma_b) \end{aligned} \quad (2.87)$$

where γ_b is the SNR per bit and R_c is the code rate. The normalized capacity, C_n and cutoff rate, R_0^* , are plotted in Fig. 2.22. From the figure we can conclude the following:



Interpretation

- (i) $R_0^* < C_n$ for all values of $\frac{E}{N_0}$. This is expected because C_n is the ultimate limit on the transmission rate R/D .

- (ii) For smaller values of $\frac{E}{N_0}$, the difference between C_n and R_0^* is approximately 3 dB. This means that randomly selected, average power limited, multi-amplitude signals yield R_0^* within 3 dB of channel capacity.

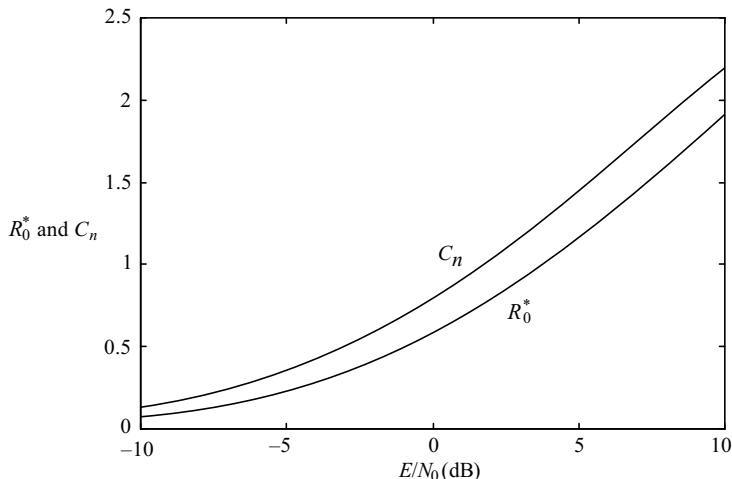


Fig. 2.22 The normalized capacity, C_n and cutoff rate, R_0^* , for an AWGN channel.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 5:

- For a BSC, the cut-off rate is given by $R_0 = 1 - \log_2 \left(1 + \sqrt{4p(1-p)}\right)$ where p is the crossover probability of the BSC.
 - For what value of p is the cut-off rate minimum? What does it physically mean?
 - Plot the cut-off rate as a function of p .
- Consider a communication system using antipodal signaling. The SNR is 20 dB.
 - Find the cutoff rate, R_0 .
 - We want to design a code which results in an average probability of error, $P_e < 10^{-6}$. What is the best code rate I can achieve?
 - What will be the dimensionality, n , of this code?
 - Repeat the earlier parts (i), (ii) and (iii) for an SNR = 5dB. Compare the results.

If you have successfully solved the above problems,
you have mastered LO 5. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/587>



2.11 Concluding Remarks

Pioneering work in the area of channel capacity was done by Shannon in 1948. Shannon's second theorem was indeed a surprising result at the time of its publication. It claimed that the probability of error for a binary symmetric channel could be made as small as desired provided the code rate was less than the channel capacity. This theorem paved way for a systematic study of reliable communication over unreliable (noisy) channels. Shannon's third theorem, the information capacity theorem, is one of the most remarkable results in information theory. It gives a relation between the channel bandwidth, the signal to noise ratio and the channel capacity. Additional work was carried out in the 1950s and 1960s by Gilbert, Gallager, Wyner, Forney and Viterbi to name some of the prominent contributors. The work related to capacity of MIMO channels was pioneered by Foschini and Telatar in the second half of 1990. The concept of cutoff rate was also developed by Shannon, but was later used by Wozencraft, Jacobs and Kennedy as a design parameter for communication systems. Jordan used the concept of cutoff rate to design coded waveforms for M -ary orthogonal signals with coherent and non-coherent detection. Cutoff rates have been widely used as a design criterion for various different channels, including fading channels encountered in wireless communications.

LEARNING OUTCOMES

- The conditional probability $P(y_i | x_j)$ is called the channel transition probability and is denoted by p_{ji} . The conditional probabilities $\{P(y_i | x_j)\}$ that characterize a DMC can be arranged in the matrix form $\mathbf{P} = [p_{ji}]$. \mathbf{P} is known as the probability transition matrix for the channel.
- The capacity of a discrete memoryless channel (DMC) is defined as the maximum average mutual information in any single use of the channel, where the maximization is overall possible input probabilities. That is,

$$C = \max_{P(x_j)} I(X; Y) = \max_{P(x_j)} \sum_{j=0}^{q-1} \sum_{i=0}^{r-1} P(x_j) P(y_i | x_j) \log \frac{P(y_i | x_j)}{P(y_i)}$$

- The basic objective of channel coding is to increase the resistance of the digital communication system to channel noise. This is done by adding redundancies in the transmitted data stream in a controlled manner. Channel coding is also referred to as error control coding.
- The ratio, $r = \frac{k}{n}$, is called the code rate. Code rate of any coding scheme is always less than unity.
- A rate r is said to be achievable if there exists a coding scheme (n, k) such that the maximal probability of error tends to 0 as $n \rightarrow \infty$. The (n, k) code may also be expressed as a $(2^nr, n)$ code or a (M, n) code, where $M = 2^k = 2^{nr}$. All rates below capacity, C , are achievable.
- A Gaussian Channel is a time discrete channel with output Y_k , at time k , which is the result of the sum of the input X_k and the Gaussian noise Z_k . This noise is drawn from a Gaussian distribution with mean zero and variance σ^2 .
- Let a DMS with an alphabet X have entropy $H(X)$ and produce symbols every T_s seconds. Let a DMC have capacity C and be used once every T_c seconds. Then, if $\frac{H(X)}{T_s} \leq \frac{C}{T_c}$, there exists a coding scheme for which the source output can be transmitted over the noisy channel and be reconstructed

with an arbitrarily low probability of error. This is the Channel Coding Theorem or the Noisy Coding Theorem.

- For $\frac{H(X)}{T_s} > \frac{C}{T_c}$, it is not possible to transmit information over the channel and reconstruct it with an arbitrarily small probability of error. The parameter is called the Critical Rate.

- The information capacity of a SISO channel can be expressed as $C = W \log_2 \left(1 + \frac{P}{N_0 W} \right)$ bits per second. This is the basic formula for the capacity of the band-limited, AWGN waveform channel with a band-limited and average power-limited input. This is the crux of the Information Capacity Theorem. This theorem is also called the Channel Capacity Theorem.

- The capacity of a frequency flat deterministic MIMO channel is given by

$$C = \max_{\text{Tr}(\mathbf{R}_{ss})=M_T} W \log_2 \det \left(\mathbf{I}_{M_R} + \frac{E_s}{M_T N_0} \mathbf{H} \mathbf{R}_{ss} \mathbf{H}^H \right) \text{ bits per second}$$

where W is the bandwidth. The condition $\text{Tr}(\mathbf{R}_{ss}) = M_T$ constrains the total average energy transmitted over a symbol period. Here the assumption is that the channel is known to the receiver.

- If the channel is unknown to the transmitter, the capacity of the MIMO channel is given by

$$C = W \sum_{i=1}^r \log_2 \left(1 + \frac{E_s}{M_T N_0} \lambda_i \right) \text{ bits per second}$$

where r is the rank of the channel and λ_i ($i = 1, 2, \dots, r$) are the positive eigen values of $\mathbf{H} \mathbf{H}^H$.

- If the channel is known to the transmitter, the different scalar data pipes may be accessed individually through processing at the transmitter and receiver. The optimal energy for each data pipe is found iteratively using the water pouring algorithm.

- The cutoff rate R_0 is given by $R_0 = \log_2 \frac{2}{1 + e^{\frac{-E}{N_0}}} = 1 - \log_2 \left(1 + e^{\frac{-E}{N_0}} \right)$. The cutoff rate has the units of

bits/dimension. Note that $0 \leq R_0 \leq 1$. The average error probability in terms of the cutoff rate can be written as $\bar{P}_e < 2^{-n(R_0 - R_c)}$. For $R_c < R_0$ the average probability of error $\bar{P}_e \rightarrow 0$ as $n \rightarrow \infty$.

Mathematics is like checkers in being suitable for the young,
not too difficult, amusing, and without peril to the state.

Plato (ca 429-347 BC)



MULTIPLE CHOICE QUESTIONS

- 2.1 The random and unpredictable electric signals from natural causes, both internal and external to the system is _____.
 (a) Interference (b) Attenuation (c) Distortion (d) Noise
- 2.2 In an ideal channel, the crossover probability, p , is
 (a) 1 (b) 0 (c) 0.5 (d) None of these
- 2.3 Relay Channels can use
 (a) Amplify-and-Forward (AF) scheme
 (b) Decode-and-Forward (DF) scheme
 (c) Hybrid of AF and DF
 (d) All of these
- 2.4 The Direct-to-Home (DTH) satellite channel is an example of
 (a) Multiple access channel (b) Broadcast channel
 (c) MIMO channel (d) None of these
- 2.5 The units of channel capacity is ‘bits per channel use’ provided the base of the logarithm is
 (a) 10 (b) 2 (c) e (d) None of these
- 2.6 For a BSC with $0.5 < p < 1$, the capacity
 (a) increases with increasing p (b) decreases with increasing p
 (c) increases with decreasing p (d) None of these
- 2.7 Channel capacity is a measure of
 (a) Entropy
 (b) Lower-bound on the maximum rate of information transfer
 (c) The maximum rate at which information can be reliably transmitted over a channel.
 (d) None of these
- 2.8 The capacity of a binary symmetric channel, given $H(p)$ is the binary entropy function, is
 (a) $1 - H(p)$ (b) $H(p) - 1$ (c) $1 - H(p)^2$ (d) $H(p)$
- 2.9 For M equally likely messages, the average amount of information H (in bits) is
 (a) $\log_{10}M$ (b) \log_2M (c) $\log_{10}M^2$ (d) $M \log_{10}M$
- 2.10 The capacity of Gaussian channel is
 (a) $C = 2W \log_2(1 + \text{SNR})$ bits/s (b) $C = W \log_{10}(1 + \text{SNR})$ bits/s
 (c) $C = W \log_2(1 + \text{SNR})$ bits/s (d) $C = W(1 + \text{SNR})$ bits/s

For interactive quiz with answers, scan the QR code given here



Or

Visit <http://qrcode.flipick.com/index.php/567>

SHORT-ANSWER TYPE QUESTIONS



- 2.1 Explain the difference between hard decision decoding and soft decision decoding.
- 2.2 Explain what is meant by a composite Discrete-input, Discrete-output channel.
- 2.3 What is a Multiple Input Multiple Output (MIMO) channel?
- 2.4 Why is the capacity of a BSC equal to zero for $p = 0.5$?
- 2.5 Suppose I have two parallel independent BSCs with crossover probabilities p and q . If I choose to send two bits each time over these parallel channels, what will be my capacity?
- 2.6 List three properties of channel capacity.
- 2.7 What is the need for channel coding?
- 2.8 What is meant by achievability of a rate?
- 2.9 Define a Gaussian Channel.
- 2.10 What are the assumptions made in the Information Capacity Theorem?
- 2.11 Explain intuitively the Water Filling algorithm.
- 2.12 What is the significance of the Shannon Limit?
- 2.13 Explain why should the capacity of a MIMO channel, with the channel known to the transmitter, be necessarily greater than or equal to the capacity when the channel is unknown to the transmitter?
- 2.14 Explain what is meant by the Capacity Region for a multiple access channel?
- 2.15 What is the physical interpretation of the Cutoff Rate?

For answers, scan the QR code given here



Or

Visit <http://qrcode.flipick.com/index.php/588>

PROBLEMS



- M 2.1 Consider the channels A, B and the cascaded channel AB shown in Fig. 2.23.

- (i) Find C_A the capacity of channel A.
- (ii) Find C_B the capacity of channel B.
- (iii) Next, cascade the two channels and determine the combined capacity C_{AB} .
- (iv) Explain the relation between C_A , C_B and C_{AB} .

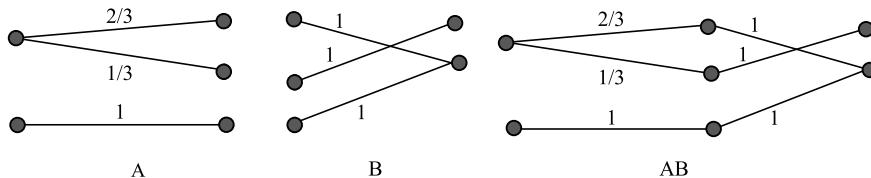


Fig. 2.23

- s** 2.2 Consider a miniature keyboard with small keys. Whenever we intend to press any key, the two adjacent keys also have equal probability of getting pressed. This is like a noisy keyboard. The equivalent channel is depicted in Fig. 2.24. Find the capacity of this noisy channel with overlapping outputs.

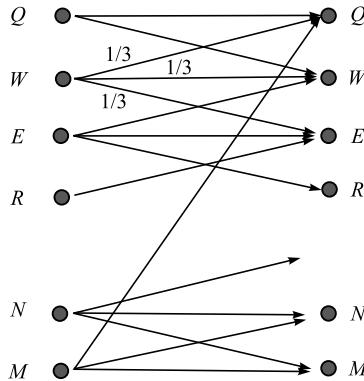


Fig. 2.24

- M** 2.3 Determine the channel capacity of the channel shown in Fig. 2.25.

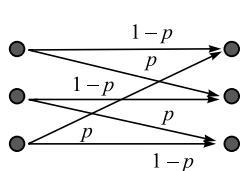


Fig. 2.25

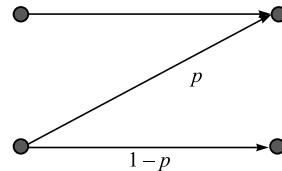


Fig. 2.26

- M** 2.4 Consider the Z channel shown Fig. 2.26.

- Find the input probabilities that result in capacity.
- If N such channels are cascaded, show that the combined channel can be represented by an equivalent Z channel with the channel transition probability $(1-p)^N$.
- What is the capacity of the combined channel as $N \rightarrow \infty$?

- M** 2.5 Consider a wireless network with n hops with each node transmitting using an orthogonal channel (Fig. 2.27). For $i = 0, 1, \dots, n-1$, node i transmits coded messages at rate R_i to node $i+1$ over an AWGN channel with noise variance σ_i^2 . Node i decodes messages of node $i-1$ and forwards to node $i+1$. Node i transmits at power P_i and the multihop system is subject to the constraint $\sum_{i=0}^{n-1} P_i = P$.

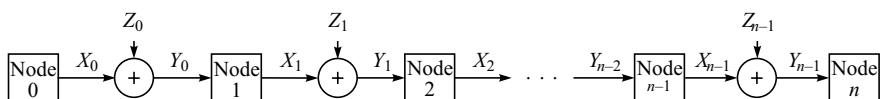


Fig. 2.27

- (i) For a given set of powers P_0, \dots, P_n , what is the capacity of C of the multihop communication system from node 0 to node n ? Express your answer in terms of C_i .

- (ii) Explain your answer in terms of the end-to-end data rate R .

- D** 2.6 Consider two binary symmetric channels with crossover probability $p \leq 1/2$ operating in parallel as shown in the Fig. 2.28. Here, for $i \in \{1, 2\}$, the i^{th} channel has input X_i , noise Z_i and output $Y_i = X_i \oplus Z_i$ where $X_i, Y_i, Z_i \in \{0, 1\}$ and where \oplus denotes modulo-two addition. The noise in the two channels is not independent and the noise pair (Z_1, Z_2) has the joint distribution $P_{Z_1, Z_2}(z_1, z_2)$ given according to the following table, in which a is a parameter satisfying $0 \leq a \leq p$.

	$z_2 = 0$	$z_2 = 1$
$z_1 = 0$	$1 + a - 2p$	$p - a$
$z_1 = 1$	$p - a$	a

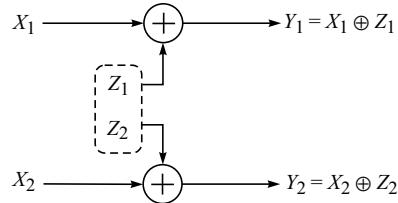


Fig. 2.28

- (i) Let p be fixed. For what value of a does the channel behave as two independent binary symmetric channels in parallel? What is the corresponding channel capacity?
(ii) Determine the channel capacity as a function of a and p .
(iii) Let p be fixed. For what value of a is the channel capacity minimized? For what value(s) of a is the channel capacity maximized? Comment on this result.

- D** 2.7 Consider a discrete-time additive white Gaussian channel subject to a periodic *Gaussian* jammer, who alternates between periods of transmission (on) and periods of non-transmission (off). The duty cycle of the jammer is $1/3$, i.e., the jammer follows the pattern on, off, off, on, off, off, ... When the jammer is off, additive noise in the channel is zero-mean Gaussian with variance $\sigma_0^2 > 0$. When the jammer is on, the additive noise is zero-mean Gaussian with variance $\sigma_1^2 > \sigma_0^2$. You have an average power constraint P , i.e., if X_1, X_2, \dots represents a sequence of channel inputs, then $\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n E[X_i^2] \leq P$.

- (i) Determine the maximum achievable rate when the transmitter is turned off when the jammer is on, i.e., $X_1 = X_4 = X_7 = \dots = 0$.
(ii) Determine the optimum transmission strategy and the corresponding maximum achievable rate of reliable information transmission.

- M** 2.8 Consider a diversity channel shown in Fig. 2.29 in which the *binary* input X is transmitted simultaneously over two independent binary symmetric channels (BSC). The BSC has a crossover probability of p .

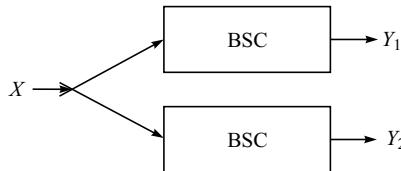


Fig. 2.29

- (i) First consider only a single BSC. Let the input probability $P_X(0)$ be equal to q . Determine the capacity of this *single* BSC.
- (ii) Next consider diversity, with two independent BSCs. Assuming $P_X(0) = 0.5$, find $I(X; Y_1, Y_2)$.
- M 2.9 Consider the discrete memoryless channel (Fig. 2.30) with source $X = \{0, 1\}$ and an independent on-off jammer, Z , such that $P(Z = 0) = P(Z = a) = 0.5$. Find the capacity of this channel.

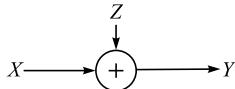


Fig. 2.30

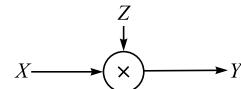


Fig. 2.31

- D 2.10 Consider the multiplier channel, shown in Fig. 2.31, with $Y = XZ$, where the source $X = \{0, 1\}$. The independent binary noise $Z = \{0, 1\}$ is Bernoulli distributed, i.e., $P(Z = 0) = 1 - a$, and $P(Z = 1) = a$. Find the capacity of this channel.
- M 2.11 Next, reconsider the multiplier channel of Prob. 2.10 with the added advantage that the receiver can observe Z , as depicted in Fig. 2.32. Find the capacity of this channel.

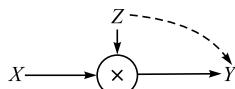


Fig. 2.32

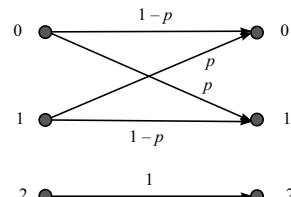


Fig. 2.33

- D 2.12 Consider a military scenario, where we have two independent BSCs at our disposal, with capacities C_1 and C_2 . In order to have an element of surprise for the enemy, sometimes BSC₁ is used, and sometimes BSC₂ is used, but not both simultaneously. Find the capacity of this ‘surprise channel’. Generalize the result if we had the luxury of K independent channels.
- S 2.13 Calculate the capacity of the ternary channel given in Fig. 2.33.
- S 2.14 Consider the fading channel, shown in Fig. 2.34, where V is a random variable representing fading and Z is the additive noise. Both V and Z are independent. Suppose we have the luxury of channel estimation, i.e., we know V . Show that the capacity with channel state information (CSI) is necessarily greater than without CSI, i.e., $I(X; Y | V) \geq I(X; Y)$.

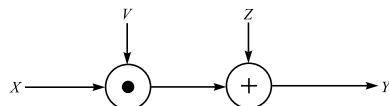


Fig. 2.34

- S 2.15 Consider a binary erasure multiple access channel with two senders transmitting either a 1 or a 0. The receiver computes $Y = X_1 + X_2$. Draw the capacity region for the binary erasure multiple access channel.

- S** 2.16 Consider a binary orthogonal broadcast channel with two independent receivers. The channel between the transmitter and receivers one and two can be modeled as a BSCs with crossover probabilities p_1 and p_2 respectively. Draw the capacity region for this orthogonal broadcast channel.

COMPUTER PROBLEMS



- 2.1 Write a computer program that takes in the channel transition probability matrix and computes the capacity of the channel.
- 2.2 Plot the operating points on the bandwidth efficiency diagram for M -PSK, $M = 2, 4, 8, 16$ and 32 , and the probabilities of error: (a) $P_e = 10^{-6}$ and (b) $P_e = 10^{-8}$.
- 2.3 Write a program that implements the binary repetition code of rate $1/n$, where n is an odd integer. Develop a decoder for the repetition code. Test the performance of this coding scheme over a BSC with the channel transition probability, p . Generalize the program for a repetition code of rate $1/n$ over $GF(q)$. Plot the residual Bit Error Rate (BER) versus p and q (make a 3-D mesh plot).
- 2.4 Write a program that can generate Capacity versus SNR plots for M_T transmit antennas and M_R receive antennas.
 - (i) Generate plots for the combinations $(M_T, M_R) = (1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (3, 1)$.
 - (ii) Compare the capacity of a MISO and a SIMO channel. Comment.
 - (iii) If we have a total number of 4 antennas, which is the best solution: $(1, 3)$, $(2, 2)$ or $(3, 1)$? Justify.

PROJECT IDEAS

- 2.1 **Cascade of Channels:** Consider a system in which information is transmitted through a series of noisy channels, with no processing allowed between successive channels. We wish to analyze a cascade of channels.
 - (i) Given a set of n channels, how should the channels be ordered so as to give maximum overall capacity?
 - (ii) How big a difference does ordering make? That is, what is the maximum difference between best and worst or best and average performance?
 - (iii) What channels will have higher capacity when cascaded with themselves n times?
 - (iv) Express the channel transition probability matrix as $P = Q\Lambda Q^{-1}$, where Q is the matrix of eigenvectors of P and Λ is the diagonal matrix of eigenvalues. Now, the n -cascade will have the transition probability matrix P^n with eigenvalues λ_i^n . Relate the channel capacity of the n -cascade with $|\lambda_i|$.
- 2.2 **Image Transmission over Binary Channels:** We wish to study how a degraded image looks after passing through binary channels. Start with a binary image (black and white image) of your choice.
 - (i) Assume that this image is transmitted through a BSC with $p = 10^{-2}$. Compare it with the original image. Repeat with $p = 10^{-1}$.
 - (ii) Assume that this image is transmitted through a Z-channel with $p = 10^{-2}$. Compare it with the original image. Repeat with $p = 10^{-1}$.

- (iii) Use the repetition code ($n = 3$) for error correction and then show the improvement in the above two cases.
- (iv) Repeat (i) and (ii) for a cascade of two channels.

2.3 Time Varying Binary Channels

- (i) Suppose the crossover probability, p , of a BSC is time varying, and is distributed uniformly over (p_1, p_2) , i.e., $p \sim U(p_1, p_2)$, $p_1 < p_2$. Find the capacity of this time varying BSC.
- (ii) Use simulations to plot the capacity versus p_1 for $p_2 = 0.5$. Can you intuitively explain this plot?
- (iii) Next, consider a Z channel with $p \sim U(p_1, p_2)$, $p_1 < p_2$. Find the capacity of this time varying Z channel.

2.4 Channel Model for Long Term Evolution (LTE)

When a wireless signal travels from a transmitter to a receiver it follows multiple paths. When these copies of the same signal arrive at the receiver they are delayed and attenuated. A well-known technique to model such a wireless channel is to use an FIR (Finite Impulse Response) filter. The wireless channel thus performs the convolution operation on the transmitted signal. Suppose, the pedestrian channel in 4G-LTE standard can be modeled using the following tap-delay line model.

Tap	1	2	3	4	5	6	7
Delay (ns)	0	30	70	90	110	190	410
Relative power (dB)	0.0	-1.0	-2.0	-3.0	-8.0	-17.2	-20.8

- (i) Simulate this LTE channel for pedestrian traffic.
- (ii) What will the received signal look like if we transmit a pulse with pulselwidth (a) 10 ns, (b) 50 ns.
- (iii) Assuming QPSK modulation, plot the BER vs. SNR curve for this channel. What role will the symbol-rate play?
- (iv) What can you say about the capacity of this channel for pedestrian traffic.

REFERENCES FOR FURTHER READING

1. Cover, Thomas M., and Joy A., Thomas. *Elements of information theory*, John Wiley & Sons, 2012.
2. Robert, Ash B., *Information Theory*. Dover Special Priced Titles, 2007.

PART - II

Error Control Coding (Channel Coding)

Chapter 3 Linear Block Codes for Error Correction

Chapter 4 Cyclic Codes

Chapter 5 Bose–Chaudhuri Hocquenghem (BCH) Codes

Chapter 6 Space–Time Codes

Linear Block Codes for Error Correction

Mathematics is an interesting intellectual sport but it should not be allowed to stand in the way of obtaining sensible information about physical processes.

Richard W. Hamming

Learning objectives

After studying this chapter the students will be able to:

- LO 1** Understand the basics of error control coding and the process of encoding linear block codes.
- LO 2** Learn how to decode linear block codes.
- LO 3** Identify some of the known good linear block codes.
- LO 4** Discuss the wonderful world of space time block codes.

3.1 Introduction to Error Correcting Codes

In this *age of information*, there is an increasing importance not only of speed, but also in accuracy of storage, retrieval, and transmission of data. The channels over which messages are transmitted are often imperfect. Machines and their non-human mistakes can turn

otherwise flawless programming into worthless, even dangerous, trash. Just as architects design buildings that will remain standing even through an earthquake, their computer counterparts have come up with sophisticated techniques capable of counteracting the digital manifestations of Murphy's Law ('If anything can go wrong, it will go'). **Error correcting codes** are a kind of safety net – the mathematical insurance against the vagaries of an imperfect material world.

Error correcting codes, as the name suggests, are used for correcting errors when messages are transmitted over a noisy channel or stored data is retrieved. The physical medium through which the messages are transmitted is called a channel (e.g., a telephone line, a satellite link, a wireless channel used for mobile

...
This chapter comes with a video overview by the author. Scan here to know more or Visit <http://qrcode.flipick.com/index.php/595>



communications etc.). Different kinds of channels are prone to different kinds of noise, which corrupt the data being transmitted. The noise could be caused by lightning, human errors, equipment malfunctioning, voltage surges, random motions of electrons in the receiver hardware, and so on. Because these error correcting codes try to overcome the detrimental effects of noise in the channel, the encoding procedure is also called *channel coding*. Error control codes are also used for accurate storage of information, for example storing data and reading it from a compact disc (CD). In this case, the error could be due to a scratch on the surface of the CD. The error correcting coding scheme will try to recover the original data from the corrupted one. The motivation for the study of error control coding comes from the fact that nowadays a large volume of data is being communicated daily and there exists an ever increasing need to communicate, store and retrieve data with greater reliability.

The basic idea behind error correcting codes is to add a certain amount of redundancy to the message prior to its transmission through the noisy channel. This redundancy, which basically consists of some extra symbols, is added in a *known* manner. The encoded message, when transmitted through the channel, might get corrupted due to noise in the channel. At the receiver, the original message can be recovered from the corrupted one if the number of errors is within the limit for which the coding strategy has been designed. By adding redundancy intelligently we can succeed in ‘fooling’ the random noise to some extent and diluting its effect. The block diagram of a digital communication system is illustrated in Fig. 3.1. Note that the most important block in the figure is that of noise, without which there will be no need for the channel encoder.

Example 3.1

Let us see how redundancy can help combat the effects of noise. The normal language that we use to communicate (say, English) has a lot of redundancy built in it already. Consider the following sentence (which may be thought of as being corrupted by noise – maybe the moths have eaten portions of a written text!):

CODNG THEORY IS AN INTRSTNG SUBJECT

As we can see, there are lots of errors in this sentence. However, due to the familiarity with the language we may probably guess that the original text would have looked something like:

CODING THEORY IS AN INTERESTING SUBJECT

We have just now used an error correcting strategy that makes use of the in-built redundancy in the English language to construct the original message from the corrupted one.

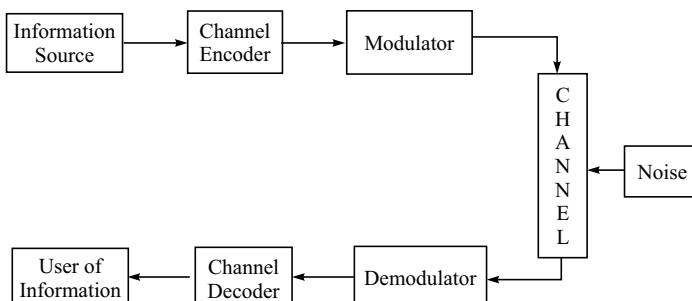


Fig. 3.1 Block diagram (and the principle) of a digital communication system. Here the source coder/decoder block has not been shown.

The objectives of a good error control coding scheme are as follows:

- (i) Error correcting capability in terms of the number of errors that it can rectify,
- (ii) Fast encoding of the message, i.e., an efficient encoding strategy,
- (iii) Fast and efficient decoding of the received message,
- (iv) Maximum transfer of information per unit time (i.e., fewer overheads in terms of the redundancy).

The first objective is the primary one. In order to increase the error correcting capability of a coding scheme one must introduce more redundancies. However, increased redundancy leads to a slower rate of transfer of the actual information. Thus the objectives (i) and (iv) are not necessarily compatible. Also, as the coding strategies become more complicated for correcting larger number of errors, the objectives (ii) and (iii) also become difficult to achieve.

In this chapter

We shall first learn about the basic definitions of error control coding. These definitions, as we shall see, would be used throughout this book. The concept of linear block codes (LBCs) will then be introduced. LBCs form a very large class of useful codes and we will see that it is very easy to work with the matrix description of these codes, in LO 1.

Next, we will learn how to efficiently decode LBCs. Specifically, we will look at syndrome decoding, in LO 2.

Subsequently, we will focus our attention on some of the known good linear block codes. We will study Hamming Codes and Low Density Parity Check (LDPC) Codes in some detail. We will also study maximum distance separable (MDS) codes and find out about optimal linear codes, in LO 3.

Finally, we will take a brief look at space time block codes (STBC). We will only scratch the surface here, as a more detailed treatment will be presented later in the book, in LO 4.

3.2 Basic Definitions

In this section we shall state some basic definitions, which will be frequently used here as well as in the later chapters.

Definition 3.1 A **word** is a sequence of symbols.

LO 1



Understand the basics of error control coding and the process of encoding linear block codes.

Definition 3.2 A **code** is a set of vectors called **codewords**.

Definition 3.3 The **Hamming weight** of a codeword (or any vector) is equal to the number of non-zero elements in the codeword. The Hamming weight of a codeword c is denoted by $w(c)$. The **Hamming distance** between two codewords is the number of places by which the codewords differ. The Hamming distance between two codewords c_1 and c_2 is denoted by $d(c_1, c_2)$. It is easy to see that $d(c_1, c_2) = w(c_1 - c_2)$.

It is easy to show that the Hamming distance follows the three properties of a distance metric which are as follows:

- (i) $d(c_1, c_2) \geq 0$ with equality if and only if $c_1 = c_2$,
- (ii) $d(c_1, c_2) = d(c_2, c_1)$.
- (iii) $d(c_1, c_2) \geq d(c_1, c_3) + d(c_2, c_3)$. This is the triangle inequality.

Example 3.2 Consider a code $C = \{0100, 1111\}$ which consists of two codewords i.e., 0100

and 1111. The Hamming weight $w(0100) = 1$ and $w(1111)$ is 4. The Hamming distance between the two codewords is 3 because they differ at the 1st, 3rd and 4th places. Observe that $w(0100 - 1111) = w(1011) = 3 = d(0100, 1111)$.

Example 3.3 For the code $C = \{01234, 43210\}$, the Hamming weight of each codeword is 4

and the Hamming distance between the codewords is 4 (because only the 3rd component of the two codewords are identical while they differ at 4 places).

Definition 3.4 A **block code** consists of a set of fixed length codewords. The fixed length of these codewords is called the **block length** and is typically denoted by n . Thus, a code of block length n consists of a set of codewords having n **components**.

A block code of size M defined over an alphabet with q symbols is a set of M q -ary sequences, each of length n . In the special case that $q = 2$, the symbols are called bits (short for **binary digits**) and the code is said to be a binary code. Usually, $M = q^k$ for some integer k , and we call such a code an (n, k) code. Thus, an **(n, k) block code** over an alphabet q is a set of q^k codewords of block length n .

Example 3.4 The code $C = \{00000, 10100, 11110, 11001\}$ is a block code of block length equal to 5. This code can be used to represent two-bit binary numbers which are as follows:

Uncoded bits	Codewords
00	00000
01	10100
10	11110
11	11001

Here $M = 4$, $k = 2$ and $n = 5$. Suppose we have to transmit a sequence of 1's and 0's using the above coding scheme. Let us say that the sequence to be encoded is 1 0 0 1 0 1 0 0 1 1....

The first step is to break the sequence in groups of two bits (because we want to encode two bits at a time). So we partition the sequence as following:

10 01 01 00 11...

Next, replace each block by its corresponding codeword.

11110 10100 10100 00000 11001...

Thus, 5 bits (coded) are sent for every 2 bits of uncoded message. It should be observed that for every 2 bits of information we are sending 3 extra bits (redundancy).

Definition 3.5 The **code rate**, r , of an (n, k) code is defined as the ratio (k/n) , and reflects the fraction of the codeword that consists of the information symbols.

Code rate is always less than unity (code rate equal to one implies no coding at all!). The smaller the code rate, the greater is the redundancy within the code, i.e., number of redundant symbols present *per information symbol* is more in a codeword. A code with greater redundancy has the potential to detect and correct greater number of symbols in error, as we shall soon see. However, a smaller code rate reduces the actual rate of transmission of information.

INDUSTRIAL RELEVANCE



As a practical case, the ITU – Optical Transport Network (OTN) recommends the use of $(255, 239)$ code with code rate 0.937. It is a non-binary code!

Definition 3.6 The **minimum distance** of a code is the minimum Hamming distance between any two codewords. If the code \mathbf{C} consists of the set of codewords $\{\mathbf{c}_i, i = 0, 1 \dots M - 1\}$ then the minimum distance of the code is given by $d^* = \min d(\mathbf{c}_i, \mathbf{c}_j), i \neq j$. An (n, k) code with minimum distance d^* is sometimes denoted by (n, k, d^*) .

Definition 3.7 The **minimum weight** of a code is the smallest weight of any non-zero codeword, and is denoted by w^* .

Definition 3.8 A **linear code** has the following properties:

- (i) The sum of two codewords belonging to the code is also a codeword belonging to the code.
- (ii) The all-zero codeword is always a codeword.
- (iii) The minimum Hamming distance between two codewords of a linear code is equal to the minimum weight of any non-zero codeword, i.e., $d^* = w^*$.

Note that if the sum of two codewords is another codeword, the difference of two codewords will also yield a valid codeword. For example, if \mathbf{c}_1 , \mathbf{c}_2 and \mathbf{c}_3 are valid codewords such that $\mathbf{c}_1 + \mathbf{c}_2 = \mathbf{c}_3$ then $\mathbf{c}_3 - \mathbf{c}_1 = \mathbf{c}_2$. Hence, it is obvious that the all-zero codeword must always be a valid codeword for a linear block code (self-subtraction of a codeword).

Theorem 3.1 For a linear code the minimum distance is equal to the minimum weight of the code, i.e., $d^* = w^*$.

Intuitive Proof: The distance d_{ij} between any two codewords c_i and c_j is simply the weight of the codeword formed by $c_i - c_j$. Since the code is linear, the difference of two codewords results in another valid codeword. Thus, the minimum weight of a non-zero codeword will reflect the minimum distance of the code.

Example 3.5 The code $\mathbf{C} = \{0000, 1010, 0101, 1111\}$ is a linear block code of block length $n = 4$. This is a $(4, 2)$ code. Observe that all the ten possible sums of the codewords

$$0000 + 0000 = 0000, 0000 + 1010 = 1010,$$

$$0000 + 0101 = 0101,$$

$$0000 + 1111 = 1111, 1010 + 1010 = 0000,$$

$$1010 + 0101 = 1111,$$

$$1010 + 1111 = 0101, 0101 + 0101 = 0000,$$

$$0101 + 1111 = 1010 \text{ and}$$

$$1111 + 1111 = 0000$$

are in \mathbf{C} and the all-zero codeword is in \mathbf{C} . The minimum distance of this code is $d^* = 2$. In order to verify the minimum distance of this linear code we can determine the distance between all pairs of codewords

(which is $\binom{4}{2} = 6$ in number):

$$d(0000, 1010) = 2, \quad d(0000, 0101) = 2, \quad d(0000, 1111) = 4$$

$$d(1010, 0101) = 4, \quad d(1010, 1111) = 2, \quad d(0101, 1111) = 2$$

We observe that the minimum distance of this code is indeed 2.

Note that the code given in Example 3.4 is not linear because $1010 + 1111 = 0101$ is not a valid codeword. Even though the all-zero word is a valid codeword, it does not guarantee linearity. *The presence of an all-zero codeword is thus a necessary but not a sufficient condition for linearity.*

Example 3.6 Consider the simple parity-check codes. Given k information bits, they add one additional parity bit so that the weight of the codeword is even. Clearly, the rate of the parity-check codes is $r = \left(\frac{k}{k+1}\right) = \left(\frac{n-1}{n}\right)$, which tends to unity as n becomes very large (i.e., for large blocklengths). For $k = 4$, we have

Information word	Codeword
0000	00000
0001	00011
0010	00101
0011	00110
...	...
...	...
1111	11110

The minimum weight $w^* = 2$ (see for example the 2nd codeword). Hence $d^* = 2$. It can detect one error and cannot correct a single error. Thus, simple parity-check codes have good rates, but poor error correcting capability.

In order to make the error correcting codes easier to use, understand and analyse, it is helpful to impose some basic algebraic structure on them. As we shall soon see, it will be useful to have an alphabet in which it is easy to carry out the basic mathematical operations such as addition, subtraction, multiplication and division.

Definition 3.9 A field \mathbf{F} is a set of elements with two operations + (addition) and . (multiplication) satisfying the following properties:

- (i) \mathbf{F} is closed under + and ., i.e., $a + b$ and $a.b$ are in \mathbf{F} if a and b are in \mathbf{F} .

For all a, b and c in \mathbf{F} , the following hold:

- (ii) Commutative laws: $a + b = b + a, a.b = b.a$
- (iii) Associative laws: $(a + b) + c = a + (b + c), a.(b.c) = (a.b).c$
- (iv) Distributive law: $a.(b + c) = a.b + a.c$.

Further, identity elements 0 and 1 must exist in \mathbf{F} satisfying:

- (v) $a + 0 = a$
- (vi) $a.1 = a$
- (vii) For any a in \mathbf{F} , there exists an additive inverse $(-a)$ such that $a + (-a) = 0$
- (viii) For any a in \mathbf{F} other than 0, there exists an multiplicative inverse (a^{-1}) such that $a(a^{-1}) = 1$.

The above properties are true for fields with both finite as well as infinite elements. A field with a finite number of elements (say, q) is called a **Galois Field** (pronounced Galva Field) and is denoted by $GF(q)$. If only the first seven properties are satisfied, then it is called a **ring**.

In a layman's language we can *loosely* define the following three things:

- (i) **Abelian group:** A set of mathematical elements that can be 'added' and 'subtracted'.
- (ii) **Ring:** A set of mathematical elements that can be 'added', 'subtracted' and 'multiplied'.
- (iii) **Field:** A set of mathematical elements that can be 'added', 'subtracted', 'multiplied' and 'divided'.

Some interesting facts about Galois Fields are listed as following:

- (i) For any Galois Field, the number of elements must be a power of prime.
- (ii) If p is a prime number and m is an integer, then there exists a Galois Field with p^m elements.
- (iii) If q is a prime power and m is an integer, then $GF(q)$ is called a subfield of $GF(q^m)$ and $GF(q^m)$ is called an extension field of $GF(q)$.
- (iv) Every Galois Field has at least one element α , called a primitive element, such that all other elements (except 0) can be expressed as a power of α .
- (v) Two Galois Fields with the same number of elements are isomorphic.

Example 3.7 Consider $GF(4)$ with 4 elements $\{0, 1, 2, 3\}$. The addition and multiplication tables for $GF(4)$ are

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

*	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

Each element has an additive inverse and a multiplicative inverse other than 0. It should be noted here that the addition in $GF(4)$ is not modulo 4 addition.

Example 3.8 Consider the following ring with 6 elements.

+	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

*	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

See, for example, elements 2, 3 or 4 do not have multiplicative inverse. Hence, it is not a Field.

Let us define a vector space, $GF(q^n)$, which is a set of n -tuples of elements from $GF(q)$. Linear block codes can be looked upon as a set of n -tuples (vectors of length n) over $GF(q)$ such that the sum of two codewords is also a codeword, and the product of any codeword by a field element is a codeword. Thus, a linear block code is a subspace of $GF(q^n)$.

Let S be a set of vectors of length n whose components are defined over $GF(q)$. The set of all linear combinations of the vectors of S is called the linear span of S and is denoted by $\langle S \rangle$. The linear span is thus a subspace of $GF(q^n)$, generated by S .

Given any subset S of $GF(q^n)$, it is possible to obtain a linear code $C = \langle S \rangle$ generated by S , consisting of precisely the following codewords:

- (i) All-zero word,
- (ii) All words in S ,
- (iii) All linear combinations of two or more words in S .

Example 3.9 Let $S = \{1100, 0100, 0011\}$. All possible linear combinations of S are

$$1100 + 0100 = 1000, \quad 1100 + 0011 = 1111, \quad 0100 + 0011 = 0111, \quad 1100 + 0100 + 0011 = 1011.$$

Therefore, $C = \langle S \rangle = \{0000, 1100, 0100, 0011, 1000, 1111, 0111, 1011\}$. The minimum distance of this code is $w(0100) = 1$.

Example 3.10 Let $S = \{12, 21\}$ defined over $GF(3)$. The addition and multiplication tables of field $GF(3) = \{0, 1, 2\}$ are given by:

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

*	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

All possible linear combinations of 12 and 21 are as follows:

$$12 + 21 = 00, \quad 12 + 2(21) = 21, \quad 2(12) + 21 = 12.$$

Therefore, $C = \langle S \rangle = \{00, 12, 21, 00, 21, 12\} = \{00, 12, 21\}$.

3.3 Matrix Description of Linear Block

LO 1

As we have observed earlier, any code C is a subspace of $GF(q^n)$. Any set of basis vectors can be used to generate the code space. We can, therefore, define a generator matrix, G , the rows of which form the basis vectors of the subspace. The rows of G will be linearly independent. Thus, a linear combination of the rows can be used to generate the codewords of C . The generator matrix will be a $k \times n$ matrix with rank k . Since the choice of the basis vectors is not unique, *the generator matrix is not unique for a given linear code*.

The generator matrix converts (encodes) a vector of length k to a vector of length n . Let the input vector (uncoded symbols) be represented by i . The coded symbols will be given by

$$c = iG \tag{3.1}$$

where c is called the codeword and i is called the information word.

The generator matrix provides a concise and efficient way of representing a linear block code. The $n \times k$ matrix can generate q^k codewords. Thus, instead of having a large look-up table of q^k codewords, one can simply have a generator matrix. This provides an enormous saving in storage space for large codes. For example, for the binary (46, 24) code the total number of codewords are $2^{24} = 1,777,216$ and the size of the lookup table of codewords will be $n \times 2^k = 771,751,936$ bits. On the other hand if we use a generator matrix, the total storage requirement would be $n \times k = 46 \times 24 = 1104$ bits.

Example 3.11 Consider a generator matrix

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

$$\mathbf{c}_1 = [0 \ 0] \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = [0 \ 0 \ 0],$$

$$\mathbf{c}_3 = [1 \ 0] \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = [1 \ 0 \ 1],$$

$$\mathbf{c}_2 = [0 \ 1] \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = [0 \ 1 \ 0]$$

$$\mathbf{c}_4 = [1 \ 1] \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = [1 \ 1 \ 1]$$

Therefore, this generator matrix generates the code $\mathbf{C} = \{000, 010, 101, 111\}$. We observe that this is a $(3, 2)$ code from the fact that the dimension of the generator matrix is 3×2 . The code rate $r = 2/3$.

3.4 Equivalent Codes

LO 1

Definition 3.10 A **permutation** of a set $S = \{x_1, x_2, \dots, x_n\}$ is a one-to-one mapping from S to itself. A permutation can be denoted as follows:

$$\begin{array}{ccccccc} x_1 & & x_2 & & \dots & & x_n \\ \downarrow & & \downarrow & & \dots & & \downarrow \\ f(x_1) & & f(x_2) & & \dots & & f(x_n) \end{array} \quad (3.2)$$

Definition 3.11 Two q -ary codes are called **equivalent** if one can be obtained from the other by one or both operations listed in the following:

- (i) Permutation of the components.
- (ii) Permutation of the position of the codeword.

Suppose a code containing M codewords are displayed in the form of an $M \times n$ matrix, where the rows represent the codewords, then operation (i) corresponds to the re-labelling of the symbols appearing in a given column, and operation (ii) represents the rearrangements of the columns of the matrix.

Example 3.12 Consider the ternary code (a code whose components $\in \{0, 1, 2\}$) of blocklength 3

$$\mathbf{C} = \begin{pmatrix} 2 & 0 & 1 \\ 1 & 2 & 0 \\ 0 & 1 & 2 \end{pmatrix}$$

If we apply the permutation $0 \rightarrow 2, 2 \rightarrow 1, 1 \rightarrow 0$ to column 2 and $1 \rightarrow 2, 0 \rightarrow 1, 2 \rightarrow 0$ to column 3 we obtain

$$\mathbf{C}_1 = \begin{pmatrix} 2 & 2 & 2 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

The code \mathbf{C}_1 is equivalent to a repetition code of length 3. Note that the original code is not linear, but is equivalent to a linear code.

Definition 3.12 Two linear q -ary codes are called **equivalent** if one can be obtained from the other by one or both operations listed below:

- (i) multiplication of the components by a non-zero scalar
- (ii) permutation of the position of the codeword.

Note that in Definition 3.11 we have defined equivalent codes that are not necessarily linear.

Theorem 3.2 Two $k \times n$ matrices generate equivalent linear (n, k) codes over $GF(q)$ if one matrix can be obtained from the other by a sequence of the following operations:

- (i) Permutation of rows
- (ii) Multiplication of a row by a non-scalar
- (iii) Addition of a scalar multiple of one row to another
- (iv) Permutation of columns
- (v) Multiplication of any column by a non-zero scalar.

Proof: The first three operations (which are just row operations) preserve the linear independence of the rows of the generator matrix. The operations merely modify the basis. The last two operations (which are column operations) convert the matrix to one which will produce an equivalent code.

Theorem 3.3 A generator matrix can be reduced to its **Systematic Form** (also called the standard form of the generator matrix) of the type $\mathbf{G} = [\mathbf{I} | \mathbf{P}]$ where \mathbf{I} is a $k \times k$ identity matrix and \mathbf{P} is a $k \times (n - k)$ matrix, called the **Parity Matrix**.

Proof: The k rows of any generator matrix (of size $k \times n$) are linearly independent. Hence, by performing elementary row operations and column permutations it is possible to obtain an equivalent generator matrix in a row echelon form. This matrix will be of the form $[\mathbf{I} | \mathbf{P}]$.

Example 3.13 Consider the generator matrix of a $(4, 3)$ code over $GF(3)$:

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

Let us represent the i^{th} row by r_i and the j^{th} column by r_j . Upon replacing r_3 by $r_3 - r_1 - r_2$ we get (note that in $GF(3)$, $-1 = 2$ and $-2 = 1$ because $1 + 2 = 0$, see table in Example 3.10)

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

Next we replace r_1 by $r_1 - r_3$ to obtain

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

Finally, shifting $c_4 \rightarrow c_1$, $c_1 \rightarrow c_2$, $c_2 \rightarrow c_3$ and $c_3 \rightarrow c_4$ we obtain the standard form of the generator matrix

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix} = \left[\begin{array}{c|cc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 2 \end{array} \right] = [\mathbf{I} | \mathbf{P}]$$

3.5 Parity Check Matrix

LO 1

One of the objectives of a good code design is to have fast and efficient encoding and decoding methodologies. So far we have dealt with the efficient generation of linear block codes using a generator matrix. Codewords are obtained simply by multiplying the input vector (uncoded word) by the generator matrix. Is it possible to detect a valid codeword using a similar concept? The answer is yes, and such a matrix is called the **parity check matrix**, \mathbf{H} , for the given code. For a parity check matrix,

$$\mathbf{c}\mathbf{H}^T = \mathbf{0} \quad (3.3)$$

where \mathbf{c} is a valid codeword. Since $\mathbf{c} = \mathbf{i}\mathbf{G}$, therefore, $\mathbf{i}\mathbf{G}\mathbf{H}^T = \mathbf{0}$. For this to hold true for all valid codewords, we must have

$$\mathbf{G}\mathbf{H}^T = \mathbf{0} \quad (3.4)$$

The size of the parity check matrix is $(n - k) \times n$. A parity check matrix provides a simple method of *detecting* whether an error has occurred or not. If the multiplication of the received word (at the receiver) with the transpose of \mathbf{H} yields a non-zero vector, it implies that an error has occurred. This methodology, however, will fail if the errors in the transmitted codeword exceeds the number of errors for which the coding scheme is designed. We shall soon find out that the non-zero product of $\mathbf{c}\mathbf{H}^T$ might help us not only to detect but also to correct the errors under some conditions.

Suppose the generator matrix is represented in its systematic form $\mathbf{G} = [\mathbf{I} | \mathbf{P}]$. The matrix \mathbf{P} is also called the **coefficient matrix**. Then the parity check matrix will be defined as:

$$\mathbf{H} = [-\mathbf{P}^T | \mathbf{I}] \quad (3.5)$$

where \mathbf{P}^T represents the transpose of matrix \mathbf{P} . This is because

$$\mathbf{G}\mathbf{H}^T = [\mathbf{I} | \mathbf{P}] \begin{bmatrix} -\mathbf{P} \\ \mathbf{I} \end{bmatrix} = \mathbf{0} \quad (3.6)$$

Since the choice of a generator matrix is not unique for a code, *the parity check matrix will not be unique*. Given a generator matrix \mathbf{G} , we can determine the corresponding parity check matrix and vice-versa. Thus the parity check matrix \mathbf{H} can be used to specify the code completely.



*Let us make the following interesting observation. Since a linear code, \mathbf{C} , is a subspace of $GF(q^n)$, it must have an orthogonal complement \mathbf{C}^\perp . Since this orthogonal complement is a subspace of $GF(q^n)$, it can also be used as a code. We call \mathbf{C}^\perp the **dual** of the code \mathbf{C} . The orthogonal complement \mathbf{C}^\perp has dimension $n - k$ and these $n - k$ basis vectors form the rows of \mathbf{H} . The orthogonality is obvious from (3.4). A code that is equal to its dual code is called a **self-dual code**. It can also be observed from (3.6) that for a self-dual code, \mathbf{P} must be a square matrix with $\mathbf{P}\mathbf{P}^T = \mathbf{I}$. It is also obvious that $(\mathbf{C}^\perp)^\perp = \mathbf{C}$.*

From (3.3) we observe that the vector c must have 1's in such positions that the corresponding rows of \mathbf{H}^T add up to the zero vector, $\mathbf{0}$. Now, we know that the number of 1's in a codeword pertains to its Hamming weight. Hence, the minimum distance d^* of a linear block code is given by the minimum number of rows of \mathbf{H}^T (or, the columns of \mathbf{H}) whose sum is equal to the zero vector. This is true for binary linear block codes. For the non-binary case, the columns of \mathbf{H} , weighted by the codewords have to be considered. Thus it is easy to determine the minimum weight of a code from the \mathbf{H} matrix: The minimum weight of a code is w^* if every set of w^*-1 columns of \mathbf{H} are linearly independent.

Example 3.14 For a $(7, 4)$ linear block code the generator matrix is given by

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

the matrix \mathbf{P} is given by $\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ and \mathbf{P}^T is given by $\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$. Observing the fact that $-1 = 1$ for

the case of binary, we can write the parity check matrix as

$$\begin{aligned} \mathbf{H} &= [-\mathbf{P}^T | \mathbf{I}] \\ &= \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

Note that the columns 1, 5 and 7 of the parity check matrix, \mathbf{H} , add up to the zero vector. Hence, for this code, $d^* = 3$.

Theorem 3.4 The code \mathcal{C} contains a non-zero codeword of Hamming weight w or less if and only if a linearly dependent set of w columns of \mathbf{H} exist.

Proof: Consider a codeword $\mathbf{c} \in \mathcal{C}$. Let the weight of \mathbf{c} be w which implies that there are w non-zero components and $(n - w)$ zero components in \mathbf{c} . If we throw away the w zero components, then from the relation $\mathbf{CH}^T = 0$ we can conclude that w columns of \mathbf{H} are linearly dependent.

Conversely, if \mathbf{H} has w linearly dependent columns, then a linear combination of most w columns is zero. These w non-zero coefficients would define a codeword of weight w or less that satisfies $\mathbf{CH}^T = 0$.

Definition 3.13 An (n, k) systematic code is one in which the first k symbols of the codeword of block length n are the information symbols themselves (i.e., the uncoded vector). The remainder of the $(n - k)$ symbols are called the **parity symbols**.

Example 3.15 The following is a $(5, 2)$ systematic code over $GF(3)$ with code rate $r = 2/5$.

S.N.	Information symbols ($k = 2$)	Codewords ($n = 5$)
1.	00	00 000
2.	01	01 121
3.	02	02 220
4.	10	10 012
5.	11	11 221
6.	12	12 210
7.	20	20 020
8.	21	21 100
9.	22	22 212

Note that the total number of codewords is $3^k = 3^2 = 9$. Each codeword begins with the information symbols, and has three parity symbols at the end. The parity symbols for the information word 01 are 121 in the above table. A generator matrix in the systematic form (standard form) will generate a systematic code.

Theorem 3.5 The minimum distance (minimum weight) of an (n, k) linear code is bounded as follows.

$$d^* \leq n - k + 1 \quad (3.7)$$



This is known as the **singleton bound**. This holds for both binary and non-binary linear block codes.

Proof: We can reduce all linear block codes to their equivalent systematic forms. A systematic code can have one information symbol and $(n - k)$ parity symbols. At most all the parity symbols can be non-zero, resulting in the total weight of the codeword to be $(n - k) + 1$. Thus the weight of no codeword can exceed $n - k + 1$.

It should be pointed out that (3.7) gives only a bound on d^* . It should not be used to determine the actual minimum distance of a code, except for a maximum distance code defined below.

Definition 3.14 A **maximum distance code** satisfies $d^* = n - k + 1$.

Having familiarised ourselves with the concept of minimum distance of a linear code, we shall now explore how this minimum distance is related to the total number of errors the code can detect and possibly correct. So we move over to the receiver's end and take a look at the methods of decoding a linear block code.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO1:

1. Consider the generator matrix $\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$. S
 - (i) Determine n and k .
 - (ii) List out all the codewords.
2. Give the addition and multiplication table for a field with two elements $\{-1, 1\}$. Can you show that it is isomorphic to $GF(2)$? S
3. Give the addition and multiplication table for $GF(5)$. Verify that all elements have an additive inverse and all non-zero elements have a multiplicative inverse. S
4. Using the $(7, 4)$ Hamming code given by $\mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$ encode the input message vectors $m_1 = 1111$ and $m_2 = 1010$. S
5. Consider $\mathbf{G}_1 = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$ and $\mathbf{G}_2 = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$. Are the two generator matrices \mathbf{G}_1 and \mathbf{G}_2 equivalent? M

If you have successfully solved the above problems,
you have mastered LO 1. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/590>



Levels of Difficulty

- S Simple:** Level 1 and Level 2 Category
- M Medium:** Level 3 and Level 4 Category
- D Difficult:** Level 5 and Level 6 Category

3.6 Decoding of a Linear Block Code

LO 2



Learn how to decode linear block codes.

The basic objective of channel coding is to detect and correct errors when messages are transmitted over a noisy channel. The noise in the channel randomly transforms some of the symbols of the transmitted codeword into some other symbols. If the noise, for example, changes just one of the symbols in the transmitted codeword, the erroneous codeword will be at a Hamming distance of one from the original codeword. If the noise transforms t symbols (that is, t symbols in the codeword are in error), the Hamming distance of the received word will be at a Hamming distance of t from the originally transmitted codeword. Given a code, how many errors can it detect and how many can it correct? Let us first look at the detection problem.

An error will be detected as long as, due to noise, one codeword does not transform into another valid codeword. If the minimum distance between the codewords is d^* , the weight of the error pattern must be d^* or more to cause a transformation from one codeword to another. Therefore, an (n, k, d^*) code will detect at least all non-zero error patterns of weight less than or equal to $(d^* - 1)$. Moreover, there is at least one error pattern of weight d^* which will not be detected. This corresponds to the two codewords that are the closest. It may be possible that some error patterns of weight d^* or more are detected, but *all* error patterns of weight d^* will not be detected.

Example 3.16 For the code $C_1 = \{000, 111\}$ the minimum distance is 3. Therefore, error patterns of weight 2 or 1 can be detected. This means that any error pattern belonging to the set $\{011, 101, 110, 001, 010, 100\}$ will be detected by this code.

Next consider the code $C_2 = \{001, 110, 101\}$ with $d^* = 1$. Nothing can be said regarding how many errors this code can detect because $d^* - 1 = 0$. However, the error pattern 010 of weight 1 can be detected by this code. But it cannot detect all error patterns with weight one, e.g., the error vector 100 cannot be detected.

Next, let us look at the problem of error correction. The objective is to make the best possible guess regarding the originally transmitted codeword after observing the received word. What could be a smart decoding strategy? Since only one of the valid codewords must have been transmitted it is logical to conclude that a valid codeword nearest (in terms of Hamming distance) to the received word must have been actually transmitted. In other words, the codeword which resembles the received word most is assumed to be the one that was sent. This strategy is called the **Nearest Neighbour decoding**, as we are picking the codeword nearest to the received word in terms of the Hamming distance.

It may be possible that more than one codeword is at the same Hamming distance from the received word. In that case the receiver can do one of the following:

- (i) It can pick one of the equally distant neighbours randomly, or
- (ii) Request the transmitter to re-transmit.

To ensure that the received word (that has at most t errors) is closest to the original codeword, and farther from all other codewords, we must put the following condition on the minimum distance of the code

$$d^* \geq 2t + 1 \quad (3.8)$$

Graphically, the condition for correcting t errors or less can be visualised from Fig. 3.2. Consider the space of all q -ary n -tuples. Every q -ary vector of length n can be represented as a point in this space. Every codeword can thus be depicted as a point in this space, and all words at a Hamming distance of t or less would

lie within the sphere centred at the codeword and with a radius of t . If the minimum distance of the code is d^* , and the condition $d^* \geq 2t + 1$ holds good, then none of these spheres would intersect. Any received vector (which is just a point) within a specific sphere will be closest to its centre (which represents a codeword) than any other codeword. We will call the spheres associated with each codeword its **Decoding Sphere**. Hence it is possible to decode the received vector using the ‘nearest neighbour’ method without ambiguity.

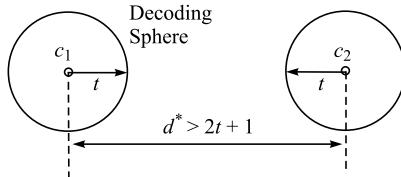


Fig. 3.2 Words within the sphere of radius t and centred at c_1 will be decoded as c_1 . For unambiguous decoding, $d^* \geq 2t + 1$.

The condition $d^* \geq 2t + 1$ takes care of the worst case scenario. It may be possible, however, that the above condition is not met but still it is feasible to correct t errors as is illustrated in the following example.

Example 3.17 Let consider the code $C = \{00000, 01010, 10101, 11111\}$. The minimum distance

$d^* = 2$. Suppose the codeword 11111 was transmitted and the received word is 11110, i.e., $t = 1$ (one error has occurred in the fifth component). Now,

$$\begin{aligned} d(11110, 00000) &= 4, d(11110, 01010) = 2 \\ d(11110, 10101) &= 3, d(11110, 11111) = 1 \end{aligned}$$

Using the nearest neighbour decoding we can conclude that 11111 was transmitted. Even though a single error correction ($t = 1$) was done in this case, $d^* < 2t + 1 = 3$. So, for certain scenarios, it is possible to correct errors even when $d^* < 2t + 1$. However, in many cases a single error correction may not be possible with this code. For example, if 00000 was sent and 01000 was received,

$$\begin{aligned} d(01000, 00000) &= 1, d(01000, 01010) = 1 \\ d(01000, 10101) &= 4, d(01000, 11111) = 4 \end{aligned}$$

In this case there cannot be a clear-cut decision, and a fair coin will have to be flipped!

Definition 3.15 An **incomplete decoder** decodes only those received codewords that are clearly closest to one of the codewords. In case of ambiguity, the decoder declares that the received word is unrecognisable. The receiver is then requested to re-transmit. A complete decoder decodes every received word, i.e., it tries to map every received word to some codeword, even if it has to make a guess.

Example 3.17 was that of a complete decoder. Such decoders may be used when it is better to have a good guess rather than to have no guess at all. Many of the real life decoders are incomplete decoders. Usually they send a message back to the transmitter requesting them to re-transmit.

Example 3.18 The **International Standard Book Number** (ISBN) is a linear block code of block length $n = 10$ over $GF(11)$. The symbols used are 0, 1, 2, ..., 9, X. Instead of using the symbol ‘10’,

'X' is used in order to avoid confusion between '10' and the case when '1' is followed by a '0'. The ISBN satisfies the following constraint:

$$\sum_{i=0}^9 (10-i)c_i = 0 \text{ calculated } (\bmod 11) \quad (3.9)$$

For example, consider the ISBN 0-07-048297-7. If we perform the check on this ISBN we get:

$$10 \times 0 + 9 \times 0 + 8 \times 7 + 7 \times 0 + 6 \times 4 + 5 \times 8 + 4 \times 2 + 3 \times 9 + 2 \times 7 + 1 \times 7 = 176 = 0 \text{ } (\bmod 11)$$

Thus 0-07-048297-7 is a valid ISBN.

Since ISBN is a linear block code, the all-zero codeword is a valid codeword. Also, 1000000001 is a valid ISBN. Thus the minimum weight of the code is $d^* = 2$. Theoretically, it cannot correct any errors and can detect a single error. However, suppose one of the digits of the ISBN gets smudged and we have 0-07-048e97-7. We can recover the erroneous digit, e , by solving $(10 \times 0 + 9 \times 0 + 8 \times 7 + 7 \times 0 + 6 \times 4 + 5 \times 8 + 4 \times e + 3 \times 9 + 2 \times 7 + 1 \times 7) \bmod 11 = 0$. The solution of this equation with one unknown yields $e = 2$. Thus, we have been able to *correct* a single error simply because we know the location of the error digit.

INDUSTRIAL RELEVANCE



ISBN is a unique numeric book identifier widely used in the publishing industry.

Definition 3.16 A receiver declares that an erasure has occurred (i.e., a received symbol has been erased) when the symbol is received ambiguously, or the presence of an interference is detected during reception.

Example 3.19 Consider a binary pulse amplitude modulation (PAM) scheme where one is represented by 5 volts and zero is represented by 0 volts. The noise margin is 1 volt, which implies that at the receiver:

- if the received voltage is between 4 volts and 5 volts \Rightarrow the bit sent is 1,
- if the received voltage is between 0 volt and 1 volts \Rightarrow the bit sent is 0,
- if the received voltage is between 1 volt and 4 volts \Rightarrow an erasure has occurred.

Thus if the receiver received 2.9 volts during a bit interval, it will declare that an erasure has occurred.

A channel can be prone both to errors and erasures. If in such a channel t errors and r erasures occur, the error correcting scheme should be able to compensate for the erasures and correct the errors as well. If r erasures occur, the minimum distance of the code will become $d^* - r$ in the worst case. This is because, the erased symbols have to be simply discarded, and if they were contributing to the minimum distance, this distance will reduce. A simple example will illustrate the point. Consider the repetition code in which

$$0 \rightarrow 00000$$

$$1 \rightarrow 11111$$

Here $d^* = 5$. If $r = 2$, i.e., two bits get erased (let us say the first two), we will have

$$0 \rightarrow ??000$$

$$1 \rightarrow ??111$$



Intuition

Now, the effective minimum distance $d_1^* = d^* - r = 3$.

Therefore, for a channel with t errors and r erasures, $d^* - r \geq 2t + 1$. Or,

$$d^* \geq 2t + r + 1 \quad (3.10)$$

For a channel which has no errors ($t = 0$), only r erasures,

$$d^* \geq r + 1. \quad (3.11)$$

Decoding in the presence of binary erasures can be done as follows. Suppose we receive a vector \mathbf{v} with r erasures. We follow the given steps:

- (i) Replace all the r erasures by '0' and carry out the regular decoding to obtain \mathbf{c}_0 . Find the Hamming distance $d_0 = d(\mathbf{c}_0, \mathbf{v})$.
- (ii) Next, replace all the r erasures by '1' and carry out the regular decoding to obtain \mathbf{c}_1 . Find the Hamming distance $d_1 = d(\mathbf{c}_1, \mathbf{v})$.
- (iii) Choose the decoded word \mathbf{c}_i corresponding to the minimum distance d_i .

Next let us give a little more formal treatment to the decoding procedure. Can we construct some mathematical tools to simplify the nearest neighbour decoding? Suppose the codeword $\mathbf{c} = c_1 c_2 \dots c_n$ is transmitted over a noisy channel. The noise in the channel changes some or all of the symbols of the codeword. Let the received vector be denoted by $\mathbf{v} = v_1 v_2 \dots v_n$. Define the **error vector** as

$$\mathbf{e} = \mathbf{v} - \mathbf{c} = v_1 v_2 \dots v_n - c_1 c_2 \dots c_n = e_1 e_2 \dots e_n \quad (3.12)$$

The decoder has to decide from the received vector, \mathbf{v} , which codeword was transmitted, or equivalently, it must determine the error vector, \mathbf{e} .

Definition 3.17 Let \mathbf{C} be an (n, k) code over $GF(q)$ and \mathbf{a} be any vector of length n . Then the set

$$\mathbf{a} + \mathbf{C} = \{\mathbf{a} + \mathbf{x} \mid \mathbf{x} \in \mathbf{C}\} \quad (3.13)$$

is called a **coset** (or translate) of \mathbf{C} . \mathbf{a} and \mathbf{b} are said to be in the same coset if $(\mathbf{a} - \mathbf{b}) \in \mathbf{C}$.

Theorem 3.6 Suppose \mathbf{C} is an (n, k) code over $GF(q)$. Then

- (i) every vector \mathbf{b} of length n is in some coset of \mathbf{C} .
- (ii) each coset contains exactly q^k vectors
- (iii) two cosets are either disjoint or coincide (partial overlap is not possible)
- (iv) if $\mathbf{a} + \mathbf{C}$ is a coset of \mathbf{C} and $\mathbf{b} \in \mathbf{a} + \mathbf{C}$, we have $\mathbf{b} + \mathbf{C} = \mathbf{a} + \mathbf{C}$.

Proofs:

- (i) $\mathbf{b} = \mathbf{b} + \mathbf{0} \in \mathbf{b} + \mathbf{C}$.
- (ii) Observe that the mapping $\mathbf{C} \rightarrow \mathbf{a} + \mathbf{C}$ defined by $\mathbf{x} \in \mathbf{a} + \mathbf{x}$, for all $\mathbf{x} \in \mathbf{C}$ is a one-to-one mapping. Thus the cardinality of $\mathbf{a} + \mathbf{C}$ is the same as that of \mathbf{C} , which is equal to q^k .
- (iii) Suppose the cosets $\mathbf{a} + \mathbf{C}$ and $\mathbf{b} + \mathbf{C}$ overlap, i.e., they have at least one vector in common. Let $\mathbf{v} \in (\mathbf{a} + \mathbf{C}) \cap (\mathbf{b} + \mathbf{C})$. Thus, for some $\mathbf{x}, \mathbf{y} \in \mathbf{C}$,

$$\mathbf{v} = \mathbf{a} + \mathbf{x} = \mathbf{b} + \mathbf{y}$$

Or, $\mathbf{b} = \mathbf{a} + \mathbf{x} - \mathbf{y} = \mathbf{a} + \mathbf{z}$, where $\mathbf{z} \in \mathbf{C}$ (because, the difference of two codewords is also a codeword).

Thus, $\mathbf{b} + \mathbf{C} = \mathbf{a} + \mathbf{C} + \mathbf{z}$ or $(\mathbf{b} + \mathbf{C}) \subset (\mathbf{a} + \mathbf{C})$.

Similarly, it can be shown that $(\mathbf{a} + \mathbf{C}) \subset (\mathbf{b} + \mathbf{C})$. From these two we can conclude that $(\mathbf{b} + \mathbf{C}) = (\mathbf{a} + \mathbf{C})$.

(iv) Since $\mathbf{b} \in \mathbf{a} + \mathbf{C}$, it implies that $\mathbf{b} = \mathbf{a} + \mathbf{x}$, for some $\mathbf{x} \in \mathbf{C}$. Next, if $\mathbf{b} + \mathbf{y} \in \mathbf{b} + \mathbf{C}$ then,

$$\mathbf{b} + \mathbf{y} = (\mathbf{a} + \mathbf{x}) + \mathbf{y} = \mathbf{a} + (\mathbf{x} + \mathbf{y}) \in \mathbf{a} + \mathbf{C}.$$

Hence, $\mathbf{b} + \mathbf{C} \subseteq \mathbf{a} + \mathbf{C}$. On the other hand, if $\mathbf{a} + \mathbf{z} \in \mathbf{a} + \mathbf{C}$, then,

$$\mathbf{a} + \mathbf{z} = (\mathbf{b} - \mathbf{x}) + \mathbf{z} = \mathbf{b} + (\mathbf{z} - \mathbf{x}) \in \mathbf{b} + \mathbf{C}.$$

Hence, $\mathbf{a} + \mathbf{C} \subseteq \mathbf{b} + \mathbf{C}$, and so $\mathbf{b} + \mathbf{C} = \mathbf{a} + \mathbf{C}$.

Definition 3.18 The vector having the minimum weight in a coset is called the **coset leader**. If there are more than one vector with the minimum weight, one of them is chosen at random and is declared the coset leader.

Example 3.20 Let \mathbf{C} be the binary $(3, 2)$ code with the generator matrix given by

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

i.e.,

$\mathbf{C} = \{000, 010, 101, 111\}$. The cosets of \mathbf{C} are

$$000 + \mathbf{C} = 000, 010, 101, 111,$$

$$001 + \mathbf{C} = 001, 011, 100, 110.$$

Note that all the eight vectors have been covered by these two cosets. As we have already seen (in the above theorem), if $\mathbf{a} + \mathbf{C}$ is a coset of \mathbf{C} and $\mathbf{b} \in \mathbf{a} + \mathbf{C}$, we have $\mathbf{b} + \mathbf{C} = \mathbf{a} + \mathbf{C}$. Hence all cosets have been listed. For the sake of illustration we write down the following

$$010 + \mathbf{C} = 010, 000, 111, 101,$$

$$011 + \mathbf{C} = 011, 001, 110, 101,$$

$$100 + \mathbf{C} = 100, 110, 001, 011,$$

$$101 + \mathbf{C} = 101, 111, 000, 101,$$

$$110 + \mathbf{C} = 110, 100, 011, 001,$$

$$111 + \mathbf{C} = 111, 101, 010, 000.$$

It can be seen that all these sets are already covered.

Since two cosets are either disjoint or coincide (from Theorem 3.6), the set of all vectors, $GF(q)^n$ can be written as

$$GF(q)^n = \mathbf{C} \cup (\mathbf{a}_1 + \mathbf{C}) \cup (\mathbf{a}_2 + \mathbf{C}) \cup \dots \cup (\mathbf{a}_t + \mathbf{C})$$

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

Definition 3.19 A **standard array** for an (n, k) code \mathbf{C} is a $q^{n-k} \times q^k$ array of *all* vectors in $GF(q)^n$ in which the first row consists of the code \mathbf{C} (with $\mathbf{0}$ on the extreme left), and the other rows are the cosets $\mathbf{a}_i + \mathbf{C}$, each arranged in corresponding order, with the coset leader on the left.

Steps for constructing a standard array:

- (i) In the first row write down all the valid codewords, starting with the all-zero codeword.

- (ii) Choose a vector \mathbf{a}_1 which is not in the first row. Write down the coset $\mathbf{a}_1 + \mathbf{C}$ as the second row such that $\mathbf{a}_1 + \mathbf{x}$ is written under $\mathbf{x} \in \mathbf{C}$.
- (iii) Next choose another vector \mathbf{a}_2 (not present in the first two rows) of minimum weight and write down the coset $\mathbf{a}_2 + \mathbf{C}$ as the third row such that $\mathbf{a}_2 + \mathbf{x}$ is written under $\mathbf{x} \in \mathbf{C}$.
- (iv) Continue the process until all the cosets are listed and every vector in $GF(q^n)$ appears exactly once.

Example 3.21 Consider the code $\mathbf{C} = \{0000, 1011, 0101, 1110\}$. The corresponding standard array is

Codewords →	0000	1011	0101	1110
	1000	0011	1101	0110
	0100	1111	0001	1010
	0010	1001	0111	1100

↑
coset leader

Note that each entry is the sum of the codeword and its coset leader.

Let us now look at the concept of decoding (obtaining the information symbols from the received codewords) using the standard array. Since the standard array comprises all possible words belonging to $GF(q^n)$, the received word can always be identified with one of the elements of the standard array. If the received word is a valid codeword, it is concluded that no errors have occurred (this conclusion may be wrong with a very low probability of error, when one valid codeword gets modified to another valid codeword due to noise!). In the case when the received word, \mathbf{v} , does not belong to the set of valid codewords, we surmise that an error has occurred. The decoder then declares that the coset leader is the error vector, \mathbf{e} , and decodes the codeword as $\mathbf{v} - \mathbf{e}$. This is the codeword at the top of the column containing \mathbf{v} . Thus, mechanically, we decode the codeword as the one on the top of the column containing the received word.

Example 3.22 Suppose the code in the previous example $\mathbf{C} = \{0000, 1011, 0101, 1110\}$ is used and the received word is $\mathbf{v} = 1101$. Since it is not one of the valid codewords, we deduce that an error has occurred. Next we try to estimate which one of the four possible codewords was actually transmitted. If we make use of the standard array of the earlier example, we find that 1101 lies in the 3rd column. The topmost entry of this column is 0101. Hence the estimated codeword is 0101. Observe that:

$$\begin{aligned} d(1101, 0000) &= 3, d(1101, 1011) = 2, \\ d(1101, 0101) &= 1, d(1101, 1110) = 2 \end{aligned}$$

and the error vector $\mathbf{e} = 1000$, the coset leader.

Codes with larger blocklengths are desirable (though not always; see the concluding remarks on this chapter) because the code rates of larger codes perform closer to the Shannon limit. As we go to larger codes (with larger values of k and n) the method of standard array will become less practical because the size of the standard array ($q^{n-k} \times q^k$) will become unmanageably large. One of the basic objectives of coding theory is to develop efficient decoding strategies. If we are to build decoders that will work in real-time, the decoding scheme should be realisable both in terms of memory required as well as the computational load. Is it possible to reduce the standard array? The answer lies in the concept of syndrome decoding, which we are going to discuss next.

3.7 Syndrome Decoding

LO 2

The standard array can be simplified if we store only the first column, and compute the remaining columns, if needed. To do so, we introduce the concept of the *syndrome* of the error pattern.

Definition 3.20 Suppose \mathbf{H} is a parity check matrix of an (n, k) code. Then for any vector $\mathbf{v} \in GF(q)^n$, the vector

$$\mathbf{s} = \mathbf{v}\mathbf{H}^T \quad (3.14)$$

is called the **syndrome** of \mathbf{v} . The syndrome of \mathbf{v} is sometimes explicitly written as $s(\mathbf{v})$. It is called a syndrome because it gives us the symptoms of the error, thereby helping us to diagnose the error. The size of the \mathbf{s} vector is $1 \times (n - k)$.

Theorem 3.7 Two vectors \mathbf{x} and \mathbf{y} are in the same coset of \mathbf{C} if and only if they have the same syndrome.

Proof: The vectors \mathbf{x} and \mathbf{y} belong to the same coset $\Leftrightarrow \mathbf{x} + \mathbf{C} = \mathbf{y} + \mathbf{C}$

$$\begin{aligned} &\Leftrightarrow \mathbf{x} - \mathbf{y} \in \mathbf{C} \\ &\Leftrightarrow (\mathbf{x} - \mathbf{y})\mathbf{H}^T = \mathbf{0} \\ &\Leftrightarrow \mathbf{x}\mathbf{H}^T = \mathbf{y}\mathbf{H}^T \\ &\Leftrightarrow s(\mathbf{x}) = s(\mathbf{y}) \end{aligned}$$

Thus there is a one-to-one correspondence between cosets and syndromes.

We can reduce the size of the standard array by simply listing the syndromes and the corresponding coset leaders.

Example 3.23 We now extend the standard array listed in Example 3.21 by adding the syndrome column.

The code is $\mathbf{C} = \{0000, 1011, 0101, 1110\}$. The corresponding standard array is

Codewords	---	0000	1011	0101	1111	00	Syndrome
	→	1000	0011	1101	0110	11	
		0100	1111	0001	1010	01	
		0010	1001	0111	1100	10	
		↑					
			Coset leader				

The steps for syndrome decoding are as follows:

- Determine the syndrome ($\mathbf{s} = \mathbf{v}\mathbf{H}^T$) of the received word, \mathbf{v} .
- Locate the syndrome in the ‘syndrome column’.
- Determine the corresponding coset leader. This is the error vector, \mathbf{e} .
- Subtract this error vector from the received word to get the codeword $\mathbf{y} = \mathbf{v} - \mathbf{e}$.

Having developed an efficient decoding methodology by means of syndrome decoding, let us now find out how much advantage does coding actually provide.

3.8 Error Probability after Coding (Probability of Error Correction)

LO 2

Definition 3.21 The **probability of error** (or, the word error rate) P_{err} for any decoding scheme is the probability that the decoder output is a wrong codeword. It is also called the **Residual Error Rate**.

Suppose there are M codewords (of length n) which are used with equal probability. Let the decoding be done using a standard array. Let the number of coset leaders with weight i be denoted by α_i . We assume that the channel is a binary symmetric channel (BSC) with symbol error probability p . A decoding error occurs if the error vector e is *not* a coset leader. Therefore, the probability of correct decoding will be

$$P_{\text{cor}} = \sum_{i=0}^n \alpha_i p^i (1-p)^{n-i} \quad (3.15)$$

Hence, the probability of error will be

$$P_{\text{err}} = 1 - \sum_{i=0}^n \alpha_i p^i (1-p)^{n-i} \quad (3.16)$$

Let us now consider m errors within the block length of n bits. We have seen that for a linear block code with minimum distance d^* , any number of errors up to $\left\lfloor \frac{1}{2}(d^* - 1) \right\rfloor$ are always correctable. The probability of m errors in a block of n bits is

$$P_{m,n} = \binom{n}{m} p^m (1-p)^{n-m} \quad (3.17)$$

where p is the probability of one bit in error. The codeword will be in error if more than t errors occur. Thus, the probability of a codeword error is upper-bounded by

$$P_M \leq \sum_{m=t+1}^n P_{m,n} \quad (3.18)$$

The equality holds for the case of **Perfect Codes**, which we shall study shortly. We also note that P_M cannot be less than the probability of erroneously decoding a transmitted codeword as another codeword which is at a distance d^* . That is

$$P_M \geq \sum_{m=\lfloor d^*/2 \rfloor + 1}^{d^*} \binom{d^*}{m} p^m (1-p)^{d^*-m} \quad (3.19)$$

We can obtain an upper-bound if we consider all the $M-1$ other codewords. All these codewords are at a distance of at least d^* . Therefore, the union bound can be expressed as

$$P_M \leq (M-1) \sum_{m=\lfloor d^*/2 \rfloor + 1}^{d^*} \binom{d^*}{m} p^m (1-p)^{d^*-m} \quad (3.20)$$

Example 3.24 Consider the standard array in Example 3.21. The coset leaders are 0000, 1000, 0100 and 0010. Therefore $\alpha_0 = 1$ (only one coset leader with weight equal to zero), $\alpha_1 = 3$ (the remaining three are of weight one) and all other $\alpha_i = 0$.

Therefore,

$$P_{err} = 1 - [(1-p)^4 + 3p(1-p)^3]$$

Recall that this code has four codewords, and can be used to send 2 bits at a time. If we did not perform coding, the probability of error of the 2-bit message being received incorrectly would be

$$P_{err} = 1 - P_{cor} = 1 - (1-p)^2$$

Note that for $p = 0.01$, the word error rate (upon coding) is $P_{err} = 0.0103$, while for the uncoded case $P_{err} = 0.0199$. So coding has almost halved the word error rate. The comparison of P_{err} for messages with and without coding is plotted in Fig. 3.3. It can be seen that coding outperforms the uncoded case only for $p < 0.5$ (guess why?). Note that the improvement due to coding comes at the cost of information transfer rate. In this example, the rate of information transfer has been cut down by half as we are sending two parity bits for every two information bits.

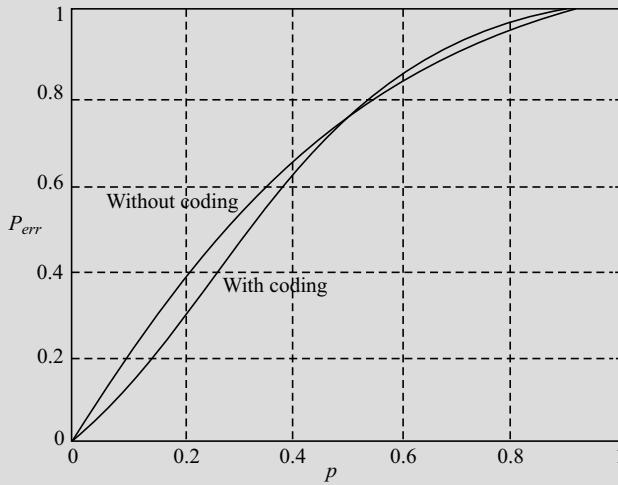


Fig. 3.3 Comparison of P_{err} for coded and uncoded 2 bit messages.

Example 3.25 This example will help us visualise the power of coding. Consider a binary symmetric channel with the probability of symbol error $p = 10^{-7}$. Suppose 10 bit long words are being transmitted *without coding*. Let the bit rate of the transmitter be 10^7 b/s, which implies that 10^6 words/s are being sent. The probability that a word is received incorrectly is

$$\binom{10}{1}(1-p)^9 p + \binom{10}{2}(1-p)^8 p^2 + \binom{10}{3}(1-p)^7 p^3 + \dots \approx \binom{10}{1}(1-p)^9 p \approx 10^{-6} \text{ words/s}$$

Therefore, in one second, $10^{-6} \times 10^6 = 1$ word will be in error! The implication is that every second a word will be in error and *it will not be detected*.

Next, let us add a bit of parity to the uncoded words so as to make them 11 bits long. The parity makes all the codewords of even and thus ensures that a single bit in error will be detected. The only way that

the coded word will be in error if two or more bits get flipped, i.e., at least two bits are in error. This can be computed as $1 - \text{probability that less than two bits are in error}$. Therefore, the probability of word error will be

$$1 - (1-p)^{11} - \binom{11}{1}(1-p)^{10}p \approx 1 - (1-11p) - 11(1-10p)p = 110p^2 = 11 \times 10^{-13}$$

The new word rate will be $10^7/11$ words/s because now 11 bits constitute one word and the bit rate is the same as before. Thus in one second, $(10^7/11) \times (11 \times 10^{-13}) = 10^{-6}$ words will be in error. This implies that, after coding, one word will be received incorrectly *without detection* every 10^6 seconds = 11.5 days! So just by increasing the word length from 10 bits (coded) to 11 bits (with coding), we have been able to obtain a dramatic decrease in the word error rate. For the second case, each time a word is detected to be in error, we can request the transmitter to re-transmit the word. This strategy for retransmission is called the **Automatic Repeat Request (ARQ)**.

INDUSTRIAL RELEVANCE



The NATO standardisation agreement STANAG 5066 uses the ARQ protocol. It is used for High Frequency (HF) Radio Data Communications.

If we have a linear block (n, k, d^*) code we can either use it for error correction or for error detection (typically combined with ARQ). Suppose we have a single error correcting code ($d^* = 3$) and two errors occur in the received word, then the syndrome in this case will be non-zero, indicating the presence of 1 or 2 errors. However, it cannot tell whether it is a single error or a double error. At this stage, two mutually exclusive actions can be taken. One strategy is to use the code as an error detecting tool and request for retransmission. In this case we do not attempt to correct the error. The other (mutually exclusive) possibility is to assume that only one error has occurred and it should be rectified. In the event that two errors have occurred, the attempt to correct a single error might introduce a third error. Thus, the word after error correction may actually contain three errors. If we try to calculate the syndrome for this wrongly ‘corrected’ word, the syndrome may turn out to be zero! This is because the code can only detect up to two errors.

It is important to ask the following question: How much energy is saved because of channel coding? The probability of error (P_e) curve versus the E_b/N_0 of a typical communication system is plotted in Fig. 3.4. It is seen from the figure that with coding, the same value of P_e can be achieved at a lower E_b/N_0 than without coding. Thus the coding scheme is able to provide a **Coding Gain**. Coding gain is measured in dB. It should be observed that the coding gain can be negative if the P_e is not low enough. This implies that channel coding provides improvement if the channel is not too bad to begin with. Also, coding gain typically increases as P_e decreases. The limiting value, as $P_e \rightarrow 0$, is called the **Asymptotic Coding Gain**.

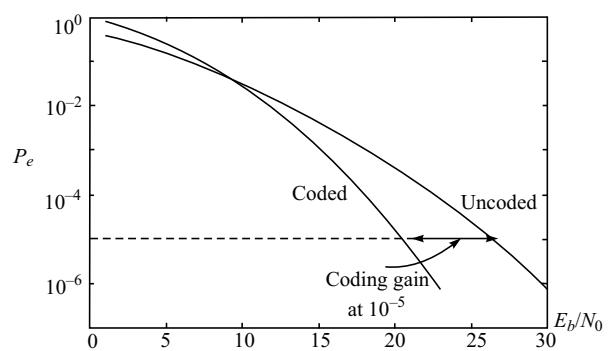


Fig. 3.4 Illustration of coding gain.



It should be pointed out that after deploying an error control code, the energy per bit goes down by a factor of the code rate. If E_b is the energy per bit for an uncoded system, the effective energy per bit after error control coding will be $E_c = (\text{code rate}) \times E_b$. This is true for all LBCs.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO2:

- Find the parity check matrix for the generator matrix $G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$. Give two realisations of the parity check matrix. S
- Consider a (15, 11) LBC with

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

- (i) How many errors can it detect?
(ii) How many errors can it correct?
(iii) How many errors can it detect in the presence of r erasures?
- Consider a (7, 1) repetition code. Let the raw error rate be $p = 0.01$. What is the probability of error after coding, i.e., the residual error rate? Can we apply this repetition code for a video transfer application over wireless requiring a BER = 10^{-7} ? M

- Consider a (7, 4) code with $G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$

Determine whether the following received words are valid codewords:

- (i) 0001101
- (ii) 0110100
- (iii) 1110000
- (iv) 1111111

If you have successfully solved the above problems,
you have mastered LO 2. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/591>



3.9 Perfect Codes

Definition 3.22 For any vector \mathbf{u} in $GF(q^n)$ and any integer $r \geq 0$, the sphere of radius r and centre \mathbf{u} , denoted by $S(\mathbf{u}, r)$, is the set $\{\mathbf{v} \in GF(q^n) \mid d(\mathbf{u}, \mathbf{v}) \leq r\}$.

LO 3



Identify some of the known good linear block codes.

This definition can be interpreted graphically, as shown in Fig. 3.5. Consider a code \mathbf{C} with minimum distance $d^*(\mathbf{C}) \geq 2t + 1$. The spheres of radius t centred at the codewords $\{c_1, c_2, \dots, c_M\}$ of \mathbf{C} will then be disjoint. Now consider the decoding problem. Any received vector can be represented as a point in this space. If this point lies within a sphere, then by nearest neighbour decoding it will be decoded as the centre of the sphere. If t or fewer errors occur, the received word will definitely lie within the sphere of the codeword that was transmitted, and will be correctly decoded. If, however, larger than t errors occur, it will escape the sphere, thus resulting in an incorrect decoding.

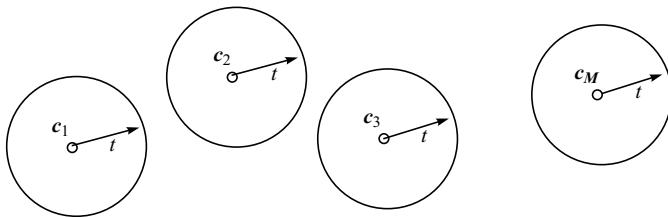


Fig. 3.5 The concept of spheres in $GF(q^n)$. The codewords of the code with $d^*(\mathbf{C}) \geq 2t + 1$ are the centres of these non-overlapping spheres.

Theorem 3.8 A sphere of radius r ($0 \leq r \leq n$) contains exactly

$$\binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \dots + \binom{n}{r}(q-1)^r \quad (3.21)$$

vectors.

Proof: Consider a vector \mathbf{u} in $GF(q^n)$ and another vector \mathbf{v} which is at a distance m from \mathbf{u} . This implies that the vectors \mathbf{u} and \mathbf{v} differ at exactly m places. The total number of ways in which m position can be chosen from n positions is $\binom{n}{m}$. Now, each of these m places can be replaced by $(q-1)$ possible symbols. This is because the total size of the alphabet is q , out of which one is currently being used in that particular position in \mathbf{u} . Hence, the number of vectors at a distance exactly m from \mathbf{u} is

$$\binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \dots + \binom{n}{r}(q-1)^r \quad (3.22)$$

Example 3.26 Consider a binary code (i.e., $q = 2$) and blocklength $n = 4$. The number of vectors at a distance 2 or less from any codeword will be

$$\binom{4}{0} + \binom{4}{1}(1) + \binom{4}{2}(1)^2 = 1 + 4 + 6 = 11$$

Without the loss of generality we can choose the fixed vector $\mathbf{u} = 0000$. The vectors of distance 2 or less are

vectors at a distance 2: 0011, 1001, 1010, 1100, 0110, 0101,

vectors at a distance 1: 0001, 0010, 0100, 1000,

vectors at a distance 0: 0000.

Thus the total number of such vectors is 11.

Theorem 3.9 A q -ary (n, k) code with M codewords and minimum distance $(2t + 1)$ satisfies

$$M \left\{ \binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \dots + \binom{n}{t}(q-1)^t \right\} \leq q^n \quad (3.23)$$

Proof: Suppose C is a q -ary (n, k) code. Consider spheres of radius t centred on the M codewords. Each sphere of radius t has

$$\binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \dots + \binom{n}{t}(q-1)^t$$

vectors (Theorem 3.8). Since none of the spheres intersect, the total number of vectors for the M disjoint spheres is $M \left\{ \binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \dots + \binom{n}{t}(q-1)^t \right\}$ which is upper bounded by q^n , the total number of vectors of length n in $GF(q)^n$.

This bound is called the **Hamming Bound** or the **Sphere Packing Bound** and it holds good for nonlinear codes as well. For binary codes, the Hamming bound will become

$$M \left\{ \binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \dots + \binom{n}{t} \right\} \leq 2^n \quad (3.24)$$

It should be noted here that just because a set of integers n, M and t satisfies the Hamming bound, the code of these specifications may not necessarily exist. For example the set $n = 5, M = 5$ and $t = 1$ satisfies the Hamming bound, however, no binary code exists for this specification.

Observe that for the case when $M = q^k$, the Hamming bound may be alternately written as:



$$\log_q \left\{ \binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \dots + \binom{n}{t}(q-1)^t \right\} \leq n - k \quad (3.25)$$

Definition 3.23 A **perfect code** is one which achieves the Hamming bound, i.e.,

$$M \left\{ \binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \dots + \binom{n}{t}(q-1)^t \right\} = q^n \quad (3.26)$$

for a perfect code.



Interpretation

For a perfect code there are equal radius disjoint spheres centred at the codewords which completely fill the space. A perfect code necessarily does not imply that it is the best possible code! A t -error correcting perfect code simply utilises the entire space in the most efficient manner.

Example 3.27 Consider the binary repetition code

$$C = \begin{cases} 00\dots0 \\ 11\dots1 \end{cases}$$

of block length n , where n is odd. In this case $M = 2$ and $t = (n - 1)/2$. Upon substituting these values in the left hand side of the inequality for Hamming bound we get

$$\text{LHS} = 2 \left\{ \binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \dots + \binom{n}{(n-1)/2} \right\} = 2 \cdot 2^{n-1} = 2^n = \text{RHS}$$

Thus the repetition code is a perfect code. It is actually called a *trivial* perfect code. In the next chapter we shall see some examples of non-trivial perfect codes.

One of the ways to search for perfect codes is to obtain the integer solutions for the parameters n , q , M and t in the equation for Hamming bound. Some of the solutions found by exhaustive computer searches are listed here.

S.N.	n	q	M	T
1	23	2	2^{12}	3
2	90	2	2^{78}	2
3	11	3	3^6	2

3.10 Hamming Codes

LO 3

There are both binary and non-binary Hamming codes. Here, we shall limit our discussion to binary Hamming codes. The binary Hamming codes have the property that



$$(n, k) = (2^m - 1, 2^m - 1 - m) \quad (3.27)$$

where m is any positive integer. For example, for $m = 3$ we have a $(7, 4)$ Hamming code. The parity check matrix, H , of a Hamming code is a very interesting matrix. Recall that the parity check matrix of an (n, k) code has $n - k$ rows and n columns. For the binary (n, k) Hamming code, the $n = 2^m - 1$ columns consist of all possible binary vectors with $n - k = m$ elements, except the all zero vector.

Example 3.28 The generator matrix for the binary (7, 4) Hamming code is given by

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

The corresponding parity check matrix is

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

Observe that the columns of the parity check matrix consist of (100), (010), (101), (110), (111), (011) and (001). These seven are all the possible non-zero binary vectors of length three. It is quite easy to generate a systematic Hamming code. The parity check matrix \mathbf{H} can be arranged in the systematic form as follows:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} = [-\mathbf{P}^T | \mathbf{I}]$$

Thus, the generator matrix in the systematic form for the binary Hamming code is

$$\mathbf{G} = [\mathbf{I} | \mathbf{P}] = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

From the above example we observe that no two columns of \mathbf{H} are linearly dependent (otherwise they would be identical). However, for $m > 1$, it is possible to identify three columns of \mathbf{H} that would add up to zero. Thus, the minimum distance, d^* , of an (n, k) Hamming code is equal to 3, which implies that it is a single-error correcting code. Hamming codes are perfect codes.

By adding an overall parity bit, an (n, k) Hamming code can be modified to yield an $(n+1, k)$ code with $d^* = 4$. On the other hand, an (n, k) Hamming code can be shortened to an $(n-l, k-l)$ code by removing l rows of its generator matrix \mathbf{G} or, equivalently, by removing l columns of its parity check matrix, \mathbf{H} . We can now give a more formal definition of Hamming Codes.

Definition 3.24 Let $n = (q^k - 1)/(q - 1)$. The (n, k) Hamming code over $GF(q)$ is a code for which the parity check matrix has columns that are pairwise linearly independent (over $GF(q)$), i.e., the columns are a maximal set of pairwise linearly independent vectors.

Example 3.29 Consider the $(8, 4)$ extended Hamming code obtained from the binary $(7, 4)$

$$\text{Hamming code } \mathbf{H}_e = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Note that we have added a row of 1's as the 1st row and introduced a column $[1 \ 0 \ 0 \dots \ 0]^T$ as the 1st column in the original \mathbf{H} matrix. The $(8, 4)$ extended Hamming code has $d^* = 4$. We further note that \mathbf{H}_e is also a generator matrix. Thus, the $(8, 4)$ extended Hamming code is a self-dual code.

In general, any binary (n, k) code with minimum distance d^* can be expanded to a $(n + 1, k)$ code with minimum distance $d^* + 1$ by appending the sum of all the codeword components to make the overall parity check. If an original codeword has an odd weight, the parity bit must be 1. Thus all codewords with weight $w^* = d^*$ become codewords of weight $d^* + 1$. Hence, the minimum weight of the code goes up by one! The

original parity check matrix, \mathbf{H} , in its expanded avatar, would become

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 0 & & & \\ \vdots & & \mathbf{H} & \\ 0 & & & \end{bmatrix}.$$

INDUSTRIAL RELEVANCE



Shortened Hamming codes have been adopted for error detection in the IEEE 802.3 Standard.

LO 3

3.11 Low Density Parity Check (LDPC) Codes

Definition 3.25 An (r, s) **Gallager code** is a linear code with a check matrix \mathbf{H} , satisfying the condition that every column has r ones and every row has s ones.

Definition 3.26 A Gallager code with small values of r and s is a **Low Density Parity Check (LDPC)** code. Thus, an LDPC code has a sparse parity check matrix, with very few ones in each row and column. Typically, $r \leq \log_2 n$, where n is the block length. The code is succinctly written as a (n, r, s) LDPC code.

Example 3.30 Consider the following parity check matrix.

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.28)$$

The matrix has $r = 2$ ones in each column and $s = 4$ ones in each row. We cannot have any more independent columns in this 5×10 matrix because all the possible combinations of choosing two ones out of the five possible locations in each column, i.e., $\binom{5}{2}$ have been exhausted. Since the dimension of the parity check matrix is $(n - k) \times n$, we have $n = 10$ and $k = 5$. We also observe that the last three columns, i.e., 8 through 10 add up to the zero vector. Therefore, the minimum distance of this code is 3. This is a $(10, 2, 4)$ LDPC code.

Definition 3.27 An LDPC code with a fixed r and s is called a **regular** LDPC code. If the number of ones in the columns and the number of ones in the rows are approximately r and s , it is called an **irregular** LDPC code.

LDPC codes can be constructed by using random constructions, algebraic constructions or a combination of the two. A simple algorithm for random construction of LDPC codes is given below.

Step 1: Set $i = 1$.

Step 2: Generate a random binary vector of length $\frac{nr}{s}$ and Hamming weight r . This is the i^{th} column of \mathbf{H} .

Step 3: If the weight of each row of \mathbf{H} at this point is $\leq s$, and the scalar product of each pair of columns is ≤ 1 , set $i = i + 1$. Else, go to step 2.

Step 4: If $i = n$, stop. Else, go to step 2.

The random algorithm does not guarantee s ones in each row. Hence, we may generate irregular LDPC codes by this method. To make the codes regular, suitable constraints have to be put in the algorithm.

An algorithm for algebraic construction of LDPC codes is given below.

Step 1: Choose $p > \frac{r-1}{s-1}$.

Step 2: Construct a $p \times p$ matrix from the identity matrix \mathbf{I}_p by cyclically shifting its rows by one position to the right.

$$\mathbf{J} = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \quad (3.29)$$

Step 3: The parity check matrix is obtained as

$$\mathbf{H} = \begin{bmatrix} \mathbf{J}^0 & \mathbf{J}^0 & \mathbf{J}^0 & \cdots & \mathbf{J}^0 \\ \mathbf{J}^0 & \mathbf{J}^1 & \mathbf{J}^2 & \cdots & \mathbf{J}^{(s-1)} \\ \mathbf{J}^0 & \mathbf{J}^2 & \mathbf{J}^3 & \cdots & \mathbf{J}^{(2(s-1))} \\ & & & \ddots & \\ \mathbf{J}^0 & \mathbf{J}^{(r-1)} & \mathbf{J}^{(2(r-1))} & \cdots & \mathbf{J}^{((r-1)(s-1))} \end{bmatrix} \quad (3.30)$$

where $\mathbf{J}^0 = \mathbf{I}_p$ and the l^{th} power of \mathbf{J} is obtained from \mathbf{I}_p by cyclically shifting its rows by $l \bmod p$ positions to the right. This matrix has exactly r ones in each column and s ones in each row. Thus it is a regular LDPC code, by design.



In a regular LDPC code, each received codeword is checked by precisely r equations and every check equation adds up precisely s code symbols. All linear codes, including LDPC codes, can be represented using a **Tanner Graph**.

Definition 3.28 A **Tanner graph** is a graphical representation of the linear code based on the set of check equations. It is a bipartite graph, which means it has two kinds of nodes: symbol nodes and check nodes. Each symbol node is connected only to check nodes and each check node is connected only to symbol nodes.

Each component of the codeword is represented in the Tanner graph by a symbol node. Hence, there are n symbol nodes. Each check equation of the code is represented in the Tanner graph by a check node. Hence, there are $n - k$ check nodes. Each check node is connected by an edge to those symbol nodes that it checks.

An LDPC code can be decoded using a simple iterative **bit flipping algorithm**. This suboptimum algorithm can be summarised in the following steps:

1. Perform hard decision decoding of the received symbols from the channel to form a binary vector \mathbf{v} . If we have the received word to start with, this step need not be performed. The vector \mathbf{v} is n bit long.
2. Compute the syndrome $\mathbf{s} = \mathbf{v}\mathbf{H}^T$. Note that each component of \mathbf{v} affects only s components of the syndrome, \mathbf{s} . If only a single bit is in error, only s syndrome components will be equal to 1.
3. Compute all check sums and the number of unsatisfied parity checks involving each of the n bits of \mathbf{v} .
4. Flip those bits of \mathbf{v} which are involved in the largest number of unsatisfied parity checks.
5. Go back to step number 3. Keep iterating until either:
 - a. All checks are satisfied, or
 - b. A predetermined number of iterations is reached.

The flipping algorithm does not guarantee that errors up to half the minimum distance are corrected. However, for large block lengths, this sub-optimum algorithm works remarkably well.

Example 3.31 Let us consider the linear rate 1/3 code given by the following parity check matrix.

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \quad (3.31)$$

Here $n = 6$, $r = 3$ and $s = 2$. This is not exactly an LDPC code since n is not large enough to yield a sparse \mathbf{H} matrix. The Tanner graph for this code is shown in Fig 3.6.

Suppose the received word is $\mathbf{v} = [0 \ 0 \ 1 \ 0 \ 0 \ 0]$. By performing syndrome calculation we obtain $\mathbf{s} = \mathbf{v}\mathbf{H}^T = [1 \ 0 \ 0 \ 1]$, which is non-zero. Hence this is not a valid codeword. From the syndrome vector we note that the parity checks that have failed are 1 and 4. This implies that there is an error among the symbols connected to the check nodes 1 and 4 of the Tanner graph. We make the following observations:

- (a) Bit 4 of the received vector corresponds to no failed checks because it is connected to check nodes 2 and 3, both of which are zeros in the syndrome vector.
- (b) Bits 1 and 2 of the received vector correspond to one failed check because they are connected to check node 1.
- (c) Bits 5 and 6 of the received vector correspond to one failed check because they are connected to check node 4.
- (d) Bit 3 of the received vector corresponds to two failed checks because it is connected to check nodes 1 and 4, both of which are ones in the syndrome vector.

According to the bit flipping algorithm, we flip the 3rd bit of the received vector to obtain [0 0 0 0 0 0]. We again perform the parity check and find that this meets all the checks. Hence, the corrected vector is the all-zero vector in this case.

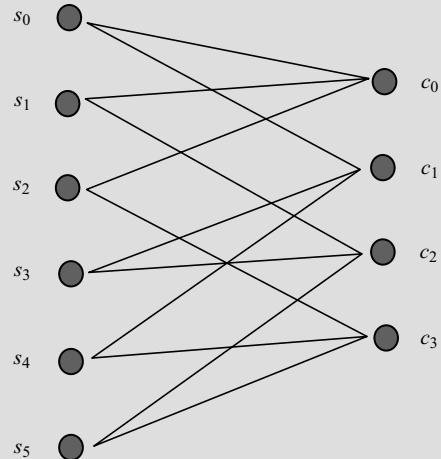


Fig. 3.6 The tanner graph corresponding to the rate 1/3 linear code discussed in Example 3.31.

INDUSTRIAL RELEVANCE



IEEE 802.16e WiMax (Worldwide Interoperability for Microwave Access) standard recommends the use of LDPC codes.

LO 3

Definition 3.29 For an (n, k, d^*) optimal code, no $(n - 1, k, d^*), (n + 1, k + 1, d^*)$ or $(n + 1, k, d^* + 1)$ code exists.

Optimal linear codes give the best distance property under the constraint of the block length. Most of the optimal codes have been found by long computer searches. It may be possible to have more than one optimal code for a given set of parameters n, k and d^* . For instance there exist two different binary $(25, 5, 10)$ optimal codes.

Example 3.32 Consider the $(24, 12, 8)$ binary code. It can be verified that

- (i) a $(23, 12, 8)$ code does not exist (only the $(23, 12, 7)$ code exists),
- (ii) a $(25, 13, 8)$ code does not exist,
- (iii) a $(25, 12, 9)$ code does not exist.

Thus the binary $(24, 12, 8)$ code is an optimal code.

LO 3

3.13 Maximum Distance Separable (MDS) Codes

In this section we consider the problem of finding as large a minimum distance d^* possible for a given redundancy, r .

Theorem 3.10 An $(n, n - r, d^*)$ code satisfies $d^* \leq r + 1$.

Proof: From the Singleton bound we have

$$d^* \leq n - k + 1$$

Substitute $k = n - r$ to get

$$d^* \leq r + 1$$

Definition 3.30 An $(n, n - r, r + 1)$ code is called a **Maximum Distance Separable (MDS)** code. An MDS code is a linear code of redundancy r , whose minimum distance is equal to $r + 1$. An MDS code meets the singleton bound.

Except the trivial repetition code, there are no binary MDS codes. However, we do have non-binary MDS codes, for example the Reed Solomon codes, which we will study in Chapter 5. Some interesting properties of MDS codes are:

- A q -ary (n, k) linear code is an MDS code if, and only if, the minimum non-zero weight of any codeword is $n - k + 1$.
- A q -ary (n, k) linear code is an MDS code if, and only if, every set of $n - k$ columns of a parity check matrix is linearly independent.

LO 3

3.14 Bounds on Minimum Distance

A simple upper bound on the minimum distance of an (n, k) binary or non-binary linear block code was given as the Singleton bound in (3.7) as $d^* \leq n - k + 1$. The normalised form of this bound can be written as

$$\frac{d^*}{n} \leq (1-r) + \frac{1}{n} \quad (3.32)$$

where r is the code rate. For large values of block length, the factor $1/n$ can be neglected. Another upper bound can be obtained directly from the Hamming bound expressed in (3.26). By dividing both sides of the inequality by n and setting $q = 2$ for the binary case, we obtain

$$\frac{1}{n} \log_2 \left(\sum_{i=0}^t \binom{n}{i} \right) \leq 1 - r \quad (3.33)$$

Since the minimum distance, d^* , and the error correcting capability, t , are related, the above relation is an upper bound on d^* . For any n , let t_0 be the largest integer t for which (3.33) holds. Then, it can be shown that as $n \rightarrow \infty$, the ratio t/n cannot exceed t_0/n , where t_0/n satisfies the equation

$$1 - r = H\left(\frac{t_0}{n}\right) \quad (3.34)$$

The generalised Hamming bound can be written as

$$\frac{1}{n} \log_q \left(\sum_{i=0}^t \binom{n}{i} (q-1)^i \right) \leq 1 - r \quad (3.35)$$

Another bound on d^* given by Plotkin Bound is

$$\frac{1}{n} \left(\frac{qd^* - 1}{q-1} - 1 - \log_q d^* \right) \leq 1 - r \quad (3.36)$$

A tighter upper bound on d^* is due to Elias, and is given below in its asymptotic form.

$$\frac{d^*}{n} \leq 2A(1-A) \quad (3.37)$$

where

$$r = 1 + A \log_2 A + (1-A) \log_2 (1-A), \quad 0 \leq A \leq \frac{1}{2} \quad (3.38)$$

Lower bounds also exist for d^* . A lower bound on d^* , based on a theorem proposed by Gilbert and Varshamov Bound is given below.

$$\frac{d^*}{n} \geq \alpha \quad (3.39)$$

where

$$r = 1 + \alpha \log_2 \alpha + (1-\alpha) \log_2 (1-\alpha) = 1 - H(\alpha), \quad 0 \leq \alpha \leq \frac{1}{2} \quad (3.40)$$

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO3:

1. Show that the binary (23, 12, 7) code is a perfect code.
2. Show that the ternary (11, 6, 7) code does not exist.
3. Consider a system without using error control coding, and allocating E_b energy per bit. What is the available energy per bit after using the LBCs given below? Also, what will be the energy loss per bit (in dB) for each of the following cases?
 - (i) (7, 4) Hamming code.
 - (ii) (15, 11) Hamming code.
 - (iii) (3, 1) Repetition code.
 - (iv) Plot the performance of the (7, 4) Hamming code over AWGN channel keeping in consideration the energy loss per bit due to coding.
4. Give the parity check matrix, H , for a (15, 11) Hamming code. From the parity check matrix determine how many errors can it correct?
5. The proposed spacecraft to Mars, *Mangalyan X*, would be sending colour photographs over a binary symmetric satellite channel that has a reliability of 0.999 and is subject to randomly scattered noises. The spacecraft creates photographs using pixels of 128 different colours. Thus each colour is a codeword. The space mission would like the probability of a pixel in the received image being assigned an incorrect colour to be less than 0.0001. Determine the parameters (n, k, d^*) of the most efficient linear code that could be used by the spacecraft.
6. Consider a linear code with distance $d^* = 3$ and length $n = 2^x$ where x is a positive integer. Use the Hamming bound to show that an upper bound U on the number of codewords in such a linear code is $U = 2^{2-x-1}$.

If you have successfully solved the above problems,
you have mastered LO 3. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/592>



3.15 Space Time Block Codes

Wireless communication involves multipath propagation, which leads to signal fading. A simple way to counter the ill-effects of fading is to use antenna diversity. However, it is not practical to deploy multiple antennas at the receiver because of size constraints. On the other hand, the base station can easily accommodate multiple antennas, thus making multiple-antenna

LO 4



Discuss the wonderful world
of space time block codes.

transmissions possible. Space-time codes are coding techniques designed for multiple-antenna transmissions. In particular, coding is performed by adding properly designed redundancy in both spatial and temporal domains. Space-time codes were first introduced by Tarokh, Seshadri, and Calderbank. In a space-time coded system, the symbol error rate (SER) for the system can be approximated by

$$\bar{P}_e \approx \frac{c}{(G_c S)^{G_d}} \quad (3.41)$$

where S is the SNR and c is a scaling constant specific to the modulation employed and the nature of the channel. $G_c (G_c \geq 1)$ denotes the **Coding Gain** and G_d is the **Diversity Order** of the system. The diversity gain/order determines the slope of an error rate curve plotted as a function of SNR, while the coding gain determines the horizontal shift of the uncoded system error rate curve to the space-time coded error rate curve obtained for the same diversity order. There are two main kinds of space-time coding schemes, namely, space-time trellis codes (STTC) and space-time block codes (STBC). While STTC are designed to provide both maximum coding and diversity gain, STBC provide only full diversity gain and zero or minimal coding gain. Here we will study the STBC and return to STTC in Chapter 8.

The **Alamouti scheme** presented in 1998 is, historically, the first space-time block code to provide full transmit diversity for systems with two transmit antennas. Tarokh later generalised Alamouti's transmit diversity scheme to an arbitrary number of transmit antennas and presented more complex STBCs based on orthogonal matrices. STBCs achieve **Maximum Likelihood** (ML) decoding through **linear processing** at the receiver. Figure 3.7 shows the block diagram of the Alamouti space-time encoder.

In order to understand the basic idea behind a space-time block code, we first look at a simple example.

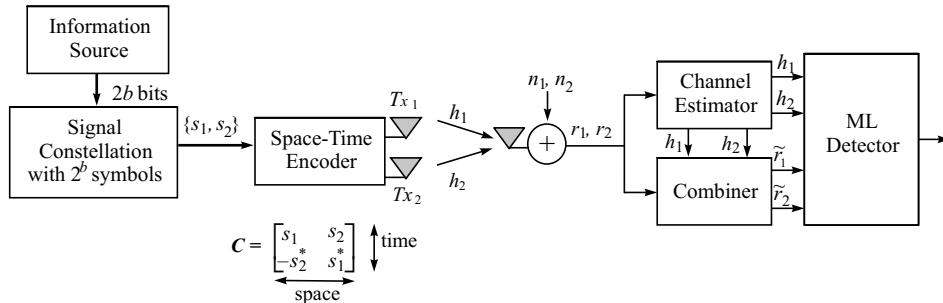


Fig. 3.7 Alamouti's two antenna transmit diversity scheme.

To transmit b bits/cycle, we use a modulation scheme that maps one symbol from a constellation with 2^b symbols. The output of the mapper, $\{x_1, x_2\}$, is input to a space-time coding block. The encoding takes in two time slots (the time axis) over two antenna elements (the space axis). In each encoding operation, the encoder takes two symbols s_1 and s_2 and transmits according to the following *coding strategy*, depicted by the matrix

$$\mathbf{X} = \begin{bmatrix} x_1 & x_2 \\ -x_2^* & x_1^* \end{bmatrix} \quad (3.42)$$

where x_1^* is the complex conjugate of x_1 . During symbol period 1, antenna 1 and antenna 2 transmit x_1 and x_2 respectively. For example, if symbols s_1 and s_2 are selected, we map $x_1 \rightarrow s_1$ and $x_2 \rightarrow s_2$ and obtain

the codeword matrix as $\mathbf{C} = \begin{bmatrix} s_1 & s_2 \\ -s_2^* & s_1^* \end{bmatrix}$.

During symbol period 1, antenna 1 and antenna 2 transmit s_1 and s_2 respectively. During symbol period 2, antenna 1 and antenna 2 transmit $-s_2^*$ and s_1^* respectively.

Note that

$$\mathbf{X}^H \mathbf{X} = \left(|x_1|^2 + |x_2|^2 \right) \mathbf{I}_2 \quad (3.43)$$

where \mathbf{I}_2 is a 2×2 identity matrix. It is assumed that the fading channel coefficients h_1 and h_2 are constant across two consecutive symbol periods.

The received signals over two consecutive symbol periods, denoted by r_1 and r_2 , can be expressed as:

$$r_1 = h_1 s_1 + h_2 s_2 + n_1 \quad (3.44)$$

$$r_2 = -h_1 s_2^* + h_2 s_1^* + n_2 \quad (3.45)$$

where n_1 and n_2 are AWGN samples for the two symbol periods respectively.

The combiner combines the received signal as follows

$$\begin{aligned} \tilde{r}_1 &= h_1^* r_1 + h_2^* r_2 \\ \tilde{r}_2 &= h_2^* r_1 - h_1^* r_2 \end{aligned} \quad (3.46)$$

Substituting (3.44) and (3.45) in (3.46) we obtain

$$\begin{aligned} \tilde{r}_1 &= \left(|h_1|^2 + |h_2|^2 \right) s_1 + h_1^* n_1 + h_2^* n_2 \\ \tilde{r}_2 &= \left(|h_1|^2 + |h_2|^2 \right) s_2 - h_1^* n_2 + h_2^* n_1 \end{aligned} \quad (3.47)$$

Note that the receiver must have an estimate of h_1 and h_2 in order to implement the combining scheme. The beauty of (3.47) is that \tilde{r}_1 depends only on s_1 , and not on s_2 . Similarly, \tilde{r}_2 depends only on s_2 , and not on s_1 . The maximum likelihood decision rule can be simplified to

$$\begin{aligned} \tilde{x}_1 &= \arg \min_{x \in S} \left| \tilde{r}_1 - \left(|h_1|^2 + |h_2|^2 \right) s \right| \\ \tilde{x}_2 &= \arg \min_{x \in S} \left| \tilde{r}_2 - \left(|h_1|^2 + |h_2|^2 \right) s \right| \end{aligned} \quad (3.48)$$

Based on the type of signal constellations, STBCs can be classified into **STBC with real signals** or **STBC with complex signals**. A **Real Orthogonal Design** of size N is an $N \times N$ matrix $\mathbf{G} = \mathbf{G}(x_1, x_2, \dots, x_N)$, with entries consisting of real elements drawn from $\pm x_1, \pm x_2, \dots, \pm x_N$ such that

$$\mathbf{G}^T \mathbf{G} = \left(\sum_{i=1}^N x_i^2 \right) \mathbf{I}_N \quad (3.49)$$

A real orthogonal design exists if and only if $N = 2, 4$ and 8 . A **Generalised Real Orthogonal Design** is a $T \times N$ matrix with entries consisting of real elements drawn from $\pm x_1, \pm x_2, \dots, \pm x_K$ such that

$$\mathbf{G}^T \mathbf{G} = \left(\sum_{i=1}^K x_i^2 \right) \mathbf{I}_N \quad (3.50)$$

where \mathbf{I}_N is a $N \times N$ identity matrix and K is a constant that represents the number of distinct symbols being transmitted. A **Complex Orthogonal Design** of size N is an $N \times N$ matrix \mathbf{G} with entries consisting of complex elements drawn from $\pm x_1, \pm x_2, \dots, \pm x_N$ and their conjugates $\pm x_1^*, \pm x_2^*, \dots, \pm x_N^*$, or multiples of these by $j = \sqrt{-1}$ such that

$$\mathbf{G}^H \mathbf{G} = \left(\sum_{i=1}^N |x_i|^2 \right) \mathbf{I}_N \quad (3.51)$$



where \mathbf{I}_N is a $N \times N$ identity matrix. A complex orthogonal design exists if and only if $N = 2$ (the Alamouti code). A **Generalised Complex Orthogonal Design** is a $T \times N$ matrix \mathbf{G} with complex elements drawn from $0, \pm x_1, \pm x_1^*, \pm x_2, \pm x_2^*, \dots, \pm x_K, \pm x_K^*$, or multiples of these by $j = \sqrt{-1}$ such that

$$\mathbf{G}^H \mathbf{G} = \kappa \left(\sum_{i=1}^K |x_i|^2 \right) \mathbf{I}_N \quad (3.52)$$

where \mathbf{I}_N is a $N \times N$ identity matrix and κ is a constant.

Some examples of real orthogonal design are

$$\mathbf{X}_2 = \begin{bmatrix} x_1 & -x_2 \\ x_2 & x_1 \end{bmatrix} \quad (3.53)$$

for $N = 2$ transmit antennas and

$$\mathbf{X}_4 = \begin{bmatrix} x_1 & -x_2 & -x_3 & -x_4 \\ x_2 & x_1 & x_4 & -x_3 \\ x_3 & -x_4 & x_1 & x_2 \\ x_4 & x_3 & -x_2 & x_1 \end{bmatrix} \quad (3.54)$$

for $N = 4$ transmit antennas.

Like in Alamouti scheme, ML decoding is accomplished in orthogonal STBCs through linear processing at the receiver because of the orthogonality of the transmission matrices. STBCs will be covered in greater detail in Chapter 6.

INDUSTRIAL RELEVANCE



The Alamouti code finds application in several modern wireless communication standards, including the IEEE 802.16e (WiMax) standard and the LTE – Advanced.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO4:

1. Show that the generator matrix, $\mathbf{G}_2 = \begin{bmatrix} x_1 & x_2 \\ -x_2 & x_1 \end{bmatrix}$, corresponds to a real orthogonal S design. What is interesting about this generator matrix?

2. Does the generator matrix, $\mathbf{G}_4 = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ -x_2 & x_1 & -x_4 & x_3 \\ -x_3 & x_4 & x_1 & -x_2 \\ -x_4 & -x_3 & x_2 & x_1 \end{bmatrix}$, correspond to a real M orthogonal? Why/why not?

3. What kind of design does the following generator matrix represent: M

$$\mathbf{G} = \begin{bmatrix} x_1 & x_2 & x_3 \\ -x_2 & x_1 & -x_4 \\ -x_3 & x_4 & x_1 \\ -x_4 & -x_3 & x_2 \end{bmatrix} \text{? What is the rate of the code?}$$

4. Is the following generator matrix a valid generalised complex orthogonal design: M

$$\mathbf{G} = \begin{bmatrix} x_1 & -x_2^* & -x_3^* & 0 \\ x_2 & x_1^* & 0 & x_3^* \\ x_3 & 0 & x_1^* & -x_2^* \\ 0 & -x_3 & x_2 & x_1 \end{bmatrix} \text{? What is interesting about this STBC scheme?}$$

If you have successfully solved the above problems,
you have mastered LO 4. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/593>



3.16 Concluding Remarks

The classic paper by Claude Elwood Shannon in the *Bell System Technical Journal* in 1948 gave birth to two important areas: (i) information theory and (ii) coding theory. At that time Shannon was only 32 years old. According to Shannon's channel coding theorem, "the error rate of data transmitted over a band-limited noisy channel can be reduced to an arbitrarily small amount if the information rate is less than the channel

capacity”. Shannon predicted the existence of good channel codes, but did not tell how to construct them. Since then the search for good codes is on. Shannon’s seminal paper can be accessed from the site[#].

In 1950, R.W. Hamming introduced the first single-error correcting code, which is still used today. The work on linear codes was extended by Golay (whose codes will be studied in the following chapter). Golay also introduced the concept of perfect codes. Non-binary Hamming codes were developed by Golay and Cocke in the late 1950s. Lately a lot of computer searches have been used to find interesting codes, however some of the best known codes are the ones discovered by sheer genius, rather than exhaustive searches.

According to Shannon’s theorem, if $C(p)$ represents the capacity (see Chapter 1 for further details) of a binary symmetric channel with probability of bit error equal to p , then for arbitrarily low probability of symbol error we must have the code rate $R < C(p)$. Even though the channel capacity provides an upperbound on the achievable code rate ($R = k/n$), evaluating a code purely against channel capacity may be misleading. The block length of the code, which translates directly into the delay, is also an important parameter. Even if a code performs far from capacity, it may be possible that it is the best code possible of the same rate and length. It has been observed that as we increase the block length of the codes, the bounds on the code rate are closer to the channel capacity as opposed to the codes with smaller blocklengths. However, longer blocklengths imply longer delays for decoding. This is because we cannot begin the decoding of a codeword until we have received the entire codeword. The maximum delay allowable is limited by practical constraints. For example, in mobile radio communications, packets of data are restricted to fewer than 200 bits. In these cases codewords with very large blocklengths cannot be used.

Low Density Parity Check (LDPC) codes were introduced by Gallager in his doctoral thesis in 1963. They were rediscovered after three decades in 1990s and have become part of several wireless standards today. The use of multiple antennas is also rapidly becoming popular in wireless communications. Transmit diversity techniques are being currently integrated with 4G wireless standards. Space time coding was introduced by Tarokh Seshadri, and Calderbank in 1988. The Alamouti scheme presented in 1998 was the first space-time block code to provide full transmit diversity for systems with two transmit antennas.

LEARNING OUTCOMES

- A word is a sequence of symbols. A code is a set of vectors called codewords.
- The Hamming weight of a codeword (or any vector) is equal to the number of non-zero elements in the codeword. The Hamming weight of a codeword c is denoted by $w(c)$.
- A block code consists of a set of fixed length codewords. The fixed length of these codewords is called the block length and is typically denoted by n . A block coding scheme converts a block of k information symbols to n coded symbols. Such a code is denoted by (n, k) .
- The code rate of an (n, k) code is defined as the ratio (k/n) , and reflects the fraction of the codeword that consists of the information symbols.
- The minimum distance of a code is the minimum Hamming distance between any two codewords. An (n, k) code with minimum distance d^* is sometimes denoted by (n, k, d^*) . The minimum weight of a code is the smallest weight of any non-zero codeword, and is denoted by w^* . For a linear code the minimum distance is equal to the minimum weight of the code, i.e., $d^* = w^*$.

[#] <http://cm.bell-labs.com/cm/ms/what/shannonday/paper.html>.

- A linear code has the following properties:
 - (i) The sum of two codewords belonging to the code is also a codeword belonging to the code.
 - (ii) The all-zero codeword is always a codeword.
 - (iii) The minimum Hamming distance between two codewords of a linear code is equal to the minimum weight of any non-zero codeword, i.e., $d^* = w^*$.
- The generator matrix converts (encodes) a vector of length k to a vector of length n . Let the input vector (uncoded symbols) be represented by \mathbf{i} . The coded symbols will be given by $\mathbf{c} = \mathbf{i}\mathbf{G}$.
- Two q -ary codes are called equivalent if one can be obtained from the other by one or both operations listed as following:
 - (i) Permutation of the components.
 - (ii) Permutation of the position of the code.
- An (n, k) systematic code is one in which the first k symbols of the codeword of block length n are the information symbols themselves. A generator matrix of the form $\mathbf{G} = [\mathbf{I} | \mathbf{P}]$ is called the systematic form or the standard form of the generator matrix, where \mathbf{I} is a $k \times k$ identity matrix and \mathbf{P} is a $k \times (n-k)$ matrix.
- The parity check matrix, \mathbf{H} , for the given code satisfies $\mathbf{c}\mathbf{H}^T = \mathbf{0}$, where \mathbf{c} is a valid codeword. Since $\mathbf{c} = \mathbf{i}\mathbf{G}$, therefore, $\mathbf{i}\mathbf{G}\mathbf{H}^T = \mathbf{0}$. The parity check matrix is not unique for a given code.
- A maximum distance code satisfies $d^* = n - k + 1$.
- For a code to be able to correct up to t errors, we must have $d^* \geq 2t + 1$, where d^* is minimum distance of the code.
- Let \mathbf{C} be an (n, k) code over $GF(q)$ and \mathbf{a} be any vector of length n . Then the set $\mathbf{a} + \mathbf{C} = \{\mathbf{a} + \mathbf{x} \mid \mathbf{x} \in \mathbf{C}\}$ is called a coset (or translate) of \mathbf{C} . \mathbf{a} and \mathbf{b} are said to be in the same coset if $(\mathbf{a} - \mathbf{b}) \in \mathbf{C}$.
- Suppose \mathbf{H} is a parity check matrix of an (n, k) code. Then for any vector $\mathbf{v} \in GF(q^n)$, the vector $\mathbf{s} = \mathbf{v}\mathbf{H}^T$ is called the syndrome of \mathbf{v} . It is called a syndrome because it gives us the symptoms of the error, thereby helping us to diagnose the error.
- A perfect code achieves the Hamming bound, i.e.,

$$M \left\{ \binom{n}{0} + \binom{n}{1} (q-1) + \binom{n}{2} (q-1)^2 + \dots + \binom{n}{t} (q-1)^t \right\} = q^n.$$

- The binary Hamming codes have the property that $(n, k) = (2^m - 1, 2^m - 1 - m)$, where m is any positive integer. Hamming codes are perfect codes.
- An (r, s) Gallager code is a linear code with a check matrix \mathbf{H} , satisfying the condition that every column has r ones and every row has s ones. Gallager code with small values of r and s is a Low Density Parity Check (LDPC) code.
- A Tanner graph is a graphical representation of the linear code based on the set of check equations. It is a bipartite graph, which means it has two kinds of nodes: symbol nodes and check nodes. Each symbol node is connected only to a check node and each check node is connected only to a symbol node.
- For an (n, k, d^*) optimal code, no $(n-1, k, d^*), (n+1, k+1, d^*)$ or $(n+1, k, d^* + 1)$ code exists.
- An $(n, n-r, r+1)$ code is called a maximum distance separable (MDS) code. An MDS code is a linear code of redundancy r , whose minimum distance is equal to $r+1$.
- Space-time codes are coding techniques designed for multiple-antenna transmissions. In particular, coding is performed by adding properly designed redundancy in both spatial and temporal domains.

- The orthogonal code matrix of the Alamouti scheme is given by $\mathbf{X} = \begin{bmatrix} x_1 & x_2 \\ -x_2^* & x_1^* \end{bmatrix}$.
- A real orthogonal design of size N is an $N \times N$ matrix $\mathbf{G} = \mathbf{G}(x_1, x_2, \dots, x_N)$, with entries consisting of real elements drawn from $\pm x_1, \pm x_2, \dots, \pm x_N$ such that $\mathbf{G}^T \mathbf{G} = \left(\sum_{i=1}^N x_i^2 \right) \mathbf{I}_N$ where \mathbf{I}_N is a $N \times N$ identity matrix.
- A generalised real orthogonal design is a $T \times N$ matrix with entries consisting of real elements drawn from $\pm x_1, \pm x_2, \dots, \pm x_\kappa$ such that $\mathbf{G}^T \mathbf{G} = \left(\sum_{i=1}^\kappa x_i^2 \right) \mathbf{I}_N$ where \mathbf{I}_N is a $N \times N$ identity matrix and κ is a constant that represents the number of distinct symbols being transmitted.
- A complex orthogonal design of size N is an $N \times N$ matrix \mathbf{G} with entries consisting of complex elements drawn from $\pm x_1, \pm x_2, \dots, \pm x_N$ and their conjugates $\pm x_1^*, \pm x_2^*, \dots, \pm x_N^*$, or multiples of these by $j = \sqrt{-1}$ such that $\mathbf{G}^H \mathbf{G} = \left(\sum_{i=1}^N |x_i|^2 \right) \mathbf{I}_N$ where \mathbf{I}_N is a $N \times N$ identity matrix.
- A generalised complex orthogonal design is a $T \times N$ matrix \mathbf{G} with complex elements drawn from $0, \pm x_1, \pm x_1^*, \pm x_2, \pm x_2^*, \dots, \pm x_\kappa, \pm x_\kappa^*$, or multiples of these by $j = \sqrt{-1}$ such that $\mathbf{G}^H \mathbf{G} = \kappa \left(\sum_{i=1}^\kappa |x_i|^2 \right) \mathbf{I}_N$ where \mathbf{I}_N is a $N \times N$ identity matrix and κ is a constant.

The greatest unsolved theorem in mathematics is why some people are better at it than others.

Adrian Mathesis



MULTIPLE CHOICE QUESTIONS

- 3.1 The code rate of an (n, k) code is given by
 - (a) k/n
 - (b) n/k
 - (c) $k/H(X)$
 - (d) None of the above
- 3.2 The d^* for a linear block code is the
 - (a) minimum Hamming distance between any two codewords
 - (b) minimum Hamming weight of a non-zero codeword
 - (c) both (a) and (b)
 - (d) None of the above
- 3.3 An (n, k) systematic code is one in which
 - (a) the last k symbols are the information symbols

- (b) the first k symbols are the information symbols
 (c) the first k symbols are the parity symbols
 (d) None of the above

3.4 A maximum distance code satisfies

- (a) $d^* = n + k + 1$
 (b) $d^* = n + k - 1$
 (c) $d^* = n - k + 1$
 (d) None of the above

3.5 For a code to be able to correct up to t errors, we must have

- (a) $d^* \geq t + 1$
 (b) $d^* \geq 2t + 1$
 (c) $d^* \geq 2t - 1$
 (d) None of the above

3.6 If \mathbf{H} is a parity check matrix of an (n, k) code, then the syndrome of the received vector, \mathbf{v} , is given by

- (a) $\mathbf{s} = \mathbf{v}\mathbf{H}^T$
 (b) $\mathbf{s} = \mathbf{v}\mathbf{H}$
 (c) $\mathbf{s} = \mathbf{v}^T\mathbf{H}$
 (d) None of the above

3.7 The code rate of repetition code is given by

- (a) $1/n$
 (b) n/k
 (c) $1/H(X)$
 (d) None of the above

3.8 Hamming codes

- (a) are perfect codes
 (b) achieve the Hamming bound
 (c) have $(n, k) = (2^m - 1, 2^m - 1 - m)$, where m is any positive integer
 (d) All of the above

3.9 For an (n, k, d^*) optimal code

- (a) no $(n - 1, k, d^*)$ code exists
 (b) no $(n + 1, k + 1, d^*)$ code exists
 (c) no $(n + 1, k, d^* + 1)$ code exists
 (d) All of the above

3.10 The code matrix of the Alamouti scheme is given by

- (a) $\mathbf{X} = \begin{bmatrix} x_1 & x_2 \\ -x_2^* & x_1^* \end{bmatrix}$
 (b) $\mathbf{X} = \begin{bmatrix} x_1 & x_2 \\ -x_2 & x_1 \end{bmatrix}$
 (c) $\mathbf{X} = \begin{bmatrix} x_1 & x_2 \\ x_2^* & x_1^* \end{bmatrix}$
 (d) $\mathbf{X} = \begin{bmatrix} x_1 & -x_2 \\ -x_2^* & x_1^* \end{bmatrix}$

For interactive quiz with answers, scan the QR code given here

Or

Visit <http://qr-code.flipick.com/index.php/568>



SHORT-ANSWER TYPE QUESTIONS

- 3.1 Explain the difference between hard decision decoding and soft decision decoding.
 3.2 How can n , k and d^* be linked?
 3.3 The code $\{11111, 22222, 33333\}$ is a $(5, 1)$ LBC over $GF(3)$. True or false?
 3.4 Find the Hamming weight of the vector $[\spadesuit 0 0 \heartsuit \clubsuit \diamondsuit 0 \clubsuit 0 \heartsuit \diamondsuit]$
 3.5 Find the Hamming distance between $\mathbf{V}_1 = [\spadesuit \clubsuit \diamondsuit \heartsuit 0 \clubsuit \diamondsuit \heartsuit]$ and $\mathbf{V}_2 = [0 \clubsuit \clubsuit 0 \clubsuit \diamondsuit \heartsuit]$
 3.6 Consider the addition and multiplication tables for $GF(2)$. What circuit elements can be used to implement them in practice?



- 3.7 Find a set of 16 binary words of length 7 such that each pair of words is at least at a Hamming distance of 3.
- 3.8 For the binary repetition code of length 5, determine the \mathbf{G} and the \mathbf{H} matrix.
- 3.9 Which of the following is not a valid Hamming code: (255, 247), (2047, 2035) and (4095, 4083)?
- 3.10 What happens if two errors occur in a received word using (7, 4) Hamming code?
- 3.11 What does a zero syndrome vector signify?
- 3.12 Define an MDS code. Is the Hamming (7, 4) code an MDS code?
- 3.13 Are all odd-length binary repetition codes perfect codes?

3.14 For the code given by $\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 2 \\ 0 & 0 & 1 & 1 & 3 \end{bmatrix}$ over $GF(4)$, what is the minimum distance, d^* ? Is this

code a perfect code?

3.15 Is the binary Hamming (15, 11) code perfect?

.....
For answers, scan the QR code given here

: Or

: Visit <http://qrcode.flipick.com/index.php/594>



PROBLEMS

- S** 3.1 Show that $\mathcal{C} = \{0000, 1100, 0011, 1111\}$ is a linear code. What is its minimum distance?
- M** 3.2 Construct, if possible, binary (n, k, d^*) codes with the following parameters:
- (i) $(6, 1, 6)$
 - (ii) $(3, 3, 1)$
 - (iii) $(4, 3, 2)$
- M** 3.3 Consider the following generator matrix over $GF(2)$.

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

- (i) Generate all possible codewords using this matrix.
- (ii) Find the parity check matrix, \mathbf{H} .
- (iii) Find the generator matrix of an equivalent systematic code.
- (iv) Construct the standard array for this code.
- (v) What is the minimum distance of this code?
- (vi) How many errors can this code detect?
- (vii) Write down the set of error patterns this code can detect.
- (viii) How many errors can this code correct?
- (ix) What is the probability of symbol error if we use this encoding scheme? Compare it with the uncoded probability of error.
- (x) Is this a linear code?

- M** 3.4 Let $\mathbf{u} = [u_1, u_2, \dots, u_n]$, $\mathbf{v} = [v_1, v_2, \dots, v_n]$ and $\mathbf{w} = [w_1, w_2, \dots, w_n]$ be binary n -tuples. Show the following:
- $d(\mathbf{u}, \mathbf{v}) = w(\mathbf{u} + \mathbf{v})$.
 - If \mathbf{u} and \mathbf{v} have even weight, $\mathbf{u} + \mathbf{v}$ has even weight.
 - $d(\mathbf{u}, \mathbf{v}) \leq d(\mathbf{u}, \mathbf{w}) + d(\mathbf{w}, \mathbf{v})$.
 - $w(\mathbf{u} + \mathbf{v}) \geq w(\mathbf{u}) - w(\mathbf{v})$.
- S** 3.5 For the code $\mathcal{C} = \{00000, 10101, 01010, 11111\}$ construct the generator matrix. Since this \mathbf{G} is not unique, suggest another generator matrix that can also generate this set of codewords.
- M** 3.6 Let $\mathbf{G}_1, \mathbf{G}_2$ be the generator matrices for two linear codes (n_1, k, d_1) and (n_2, k, d_2) respectively.
- What are the three parameters (n, k, d) for the code with $\mathbf{G} = [\mathbf{G}_1 | \mathbf{G}_2]$?
 - What are the three parameters (n, k, d) for the code with $\mathbf{G} = \begin{bmatrix} \mathbf{G}_1 & 0 \\ 0 & \mathbf{G}_2 \end{bmatrix}$?
- S** 3.7 Show that if there is a binary (n, k, d) code with d^* even, then there exists a binary (n, k, d^*) code in which all codewords have even weight.
- S** 3.8 Show that if \mathcal{C} is a binary linear code, then the code obtained by adding an overall parity check bit to \mathcal{C} is also linear.
- S** 3.9 Find the parity check matrix, \mathbf{H} , for an $(n, 8, 4)$ binary linear code with minimum n .
- S** 3.10 Let us denote vectors $\mathbf{u} = u_1 u_2 u_3 \dots u_n$ and $\mathbf{v} = v_1 v_2 v_3 \dots v_n$. Denote $(\mathbf{u}|\mathbf{v}) = u_1 u_2 u_3 \dots u_n v_1 v_2 v_3 \dots v_n$ as a vector of length $2n$. Next, suppose \mathcal{C}_1 is a binary (n, k_1, d_1) code and \mathcal{C}_2 is a binary (n, k_2, d_2) code with codewords $\mathbf{u} \in \mathcal{C}_1$ and $\mathbf{v} \in \mathcal{C}_2$, where d_1 and d_2 represent the minimum distances of the codes respectively. Form a new codeword \mathbf{c}_3 as $(\mathbf{u} | \mathbf{u} + \mathbf{v})$.
 - Show that \mathcal{C}_3 is a $(2n, k_1 + k_2)$ code.
 - What is the minimum distance of \mathcal{C}_3 ?
- S** 3.11 For each of the following sets \mathcal{S} , list the code $\langle \mathcal{S} \rangle$.
- $\mathcal{S} = \{0101, 1010, 1100\}$.
 - $\mathcal{S} = \{1000, 0100, 0010, 0001\}$.
 - $\mathcal{S} = \{11000, 01111, 11110, 01010\}$.
- S** 3.12 Consider the $(23, 12, 7)$ binary code. Show that if its used over a binary symmetric channel (BSC) with probability of bit error $p = 0.01$, the word error will be approximately 0.00008.
- M** 3.13 Suppose \mathcal{C} is a binary code with parity check matrix, \mathbf{H} . Show that the extended code \mathcal{C}_1 , obtained from \mathcal{C} by adding an overall parity bit, has the parity check matrix.

$$\mathbf{H}_1 = \left[\begin{array}{cccccc|c} & & & & & & 0 \\ & & & & & & 0 \\ & & & & & & 0 \\ \mathbf{H} & & & & & & \\ \hline & & & & & & \\ - & - & - & - & - & - & - \\ 1 & 1 & \dots & 1 & | & 1 \end{array} \right]$$

- D** 3.14 For a $(5, 3)$ code over $GF(4)$, the generator matrix is given by

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 2 \\ 0 & 0 & 1 & 1 & 3 \end{bmatrix}$$

- (i) Find the parity check matrix.
- (ii) How many errors can this code detect?
- (iii) How many errors can this code correct?
- (iv) How many erasures can this code correct?
- (v) Is this a perfect code?

s 3.15 Consider a linear block code over $GF(11)$ with blocklength $n = 10$, satisfying the following two constraints: $\sum_{i=0}^9 i c_i = 0$ calculated $(\bmod \ 11)$ and $\sum_{i=0}^9 (10-i)c_i = 0$ calculated $(\bmod \ 11)$. Find the minimum distance of this code.

D 3.16 Consider the generator matrix over $GF(5)$, $\mathbf{G} = \begin{bmatrix} 2 & 4 & 2 & 2 & 4 & 4 \\ 0 & 0 & 3 & 0 & 1 & 1 \\ 3 & 1 & 4 & 0 & 4 & 0 \\ 2 & 3 & 4 & 1 & 1 & 1 \end{bmatrix}$. Find the parity check matrix, \mathbf{H} .

- M** 3.17 Let \mathbf{C} be a binary perfect code of length n with minimum distance 7. Show that $n = 7$ or $n = 23$.
- s** 3.18 Consider a linear binary code \mathbf{C} and a vector $\mathbf{u} \notin \mathbf{C}$. Show that $\mathbf{C} \cup (\mathbf{u} + \mathbf{C})$ is also a linear code.
- S** 3.19 Let r_H denote the code rate for the binary Hamming code. Determine $\lim_{k \rightarrow \infty} r_H$.
- s** 3.20 Show that a $(15, 8, 5)$ code does not exist.
- S** 3.21 Does a ternary double-error correcting perfect code exist? Explain.
- M** 3.22 Check whether the following STBC code is orthogonal:

$$\mathbf{G} = \begin{bmatrix} x_1 & -x_2^* & -x_3^* & 0 \\ x_2 & x_1^* & 0 & -x_3^* \\ x_3 & 0 & x_1^* & x_2^* \end{bmatrix}$$

- M** 3.23 Check whether the following STBC code is orthogonal:

$$\mathbf{G} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ -x_2 & x_1 & -x_4 & x_3 \\ -x_3 & x_4 & x_1 & -x_2 \\ -x_4 & -x_3 & x_2 & x_1 \\ x_1^* & x_2^* & x_3^* & x_4^* \\ -x_2^* & x_1^* & -x_4^* & x_3^* \\ -x_3^* & x_4^* & x_1^* & -x_2^* \\ -x_4^* & -x_3^* & x_2^* & x_1^* \end{bmatrix}$$

COMPUTER PROBLEMS



- 3.1 Write a computer program to find the minimum distance of a linear block code over $GF(2)$, given the generator matrix for the code.
- 3.2 Generalise the above program to find the minimum distance of any linear block code over $GF(q)$.
- 3.3 Show by simulations that if a binary (15, 11) Hamming code is used on a channel that makes two errors then the output of the decoder is always wrong. What can you say about $(2^m - 1, 2^m - 1 - m)$ Hamming code in general?
- 3.4 Write a computer program to exhaustively search for all the perfect code parameters n, q, M and t in the equation for the Hamming bound. Search for $1 \geq n \geq 200, 2 \geq q \geq 11$.
- 3.5 Write a computer program for a universal binary Hamming encoder with rate $\frac{2^m - 1}{2^m - 1 - m}$. The program should take as input the value of m and a bit-stream to be encoded. It should then generate an encoded bit-stream. Develop a program for the decoder also. Now, perform the following tasks.
- (i) Write an error generator module that takes in a bit stream and outputs another bit-stream after inverting every bit with probability p , i.e., the probability of a bit error is p .
 - (ii) For $m = 3$, pass the Hamming encoded bit-stream through the above-mentioned module and then decode the received words using the decoder block.
 - (iii) Plot the residual error probability (the probability of error after decoding) as a function of p . Note that if you are working in the range of $BER = 10^{-r}$, you must transmit on the order of 10^{r+2} bits (why?).
 - (iv) Repeat your simulations for $m = 5, 8$ and 15 . What happens as $m \rightarrow \infty$.
- 3.6 Write a computer program to generate a (20, 3, 4) LDPC code using:
- (i) the random construction method,
 - (ii) the algebraic construction method.
- Compare the two.
- 3.7 Write a computer program to generate a table of Hamming distances between all pairs of codewords for a given Hamming code. The table would be of size $(2^k + 1) \times (2^k + 1)$. The 1st row and 1st column should list out all the valid codewords. All the other entries in this table will be Hamming distances. Specifically, carry out this exercise for the (7, 4) Hamming code.
- 3.8 Generate another table of size $(2^k + 1) \times (2^k + 1)$ which shows that sum of each pair of codeword also results in another valid codeword. All the entries in this table will be codewords. Specifically carry out this exercise for the (15, 11) Hamming code.
- 3.9 You are given 12 gold coins, of which 1 is a counterfeit (weighs either more or less than a genuine gold coin). You have a beam balance at your disposal. Write a program that uses the parity-check matrix of a (12, 9) LBC over $GF(3)$ in order to detect the counterfeit coin with three weighings.
- 3.10 Write a computer program to search for orthogonal STBCs with $N > 2$.

PROJECT IDEAS

- 3.1 **A communication System with LBC:** Simulate a wireless communication system consisting of a Channel Encoder, a Fading Channel and a Decoder. The aim of the project is to evaluate the performance of an $(2^m - 1, 2^m - 1 - m)$ Hamming Code over a slow-fading Rayleigh channel. Specifically, provide the following for a range of SNRs:

- (i) probability of word error,
- (ii) probability of bit error,
- (iii) probability of undetected codeword error.

- 3.2 **Concatenated Codes:** We would like to study the behavior of concatenated codes, and whether the sequence of coding makes a difference. Simulate a system consisting of a concatenated encoder, an AWGN Channel and a concatenated decoder. The concatenated encoder consists of two or more LBCs with parameters (n_i, k_i) , $i = 1, 2, \dots, p$. Similarly, the concatenated decoder consists of a series of decoders. Determine the BER performance of the concatenated codes. Does the sequence of the block codes make a difference? Explain.
- 3.3 **Circuit Implementation:** We would like to implement a general Hamming code using basic circuit elements. Use shift-registers and xor gates to implement a $(2^m - 1, 2^m - 1 - m)$ Hamming Code. How will you implement non-binary Hamming codes using these circuit elements?
- 3.4 **Extended Hamming Code:** Use an over-all parity check bit on the $(15, 11)$ Hamming code to convert it into a $(16, 11)$ code that can (i) detect all double errors and (ii) correct all single errors. Will adding 2 parity bits buy you anything? Does the $(16, 11)$ code attain the Plotkin bound? Also, derive an expression for the probability of decoding error when this $(16, 11)$ code is used over a BSC with crossover probability, p .
- 3.5 **Weight Distribution:** For an (n, k) code \mathcal{C} , defined over $GF(q)$, let A_i be the number of codewords with weight i . Define the weight enumerator polynomial

$$A(z) = A_0 + A_1 z + A_2 z^2 + \dots + A_n z^n$$

which essentially gives the weight distribution. If $B(z)$ is the weight enumerator polynomial for the dual code \mathcal{C}^\perp , then show that

$$B(z) = q^{-k} (1 + (q-1)z)^n A\left(\frac{1-z}{1-(q-1)z}\right)$$

Write a computer program that gives the weight distribution of any (n, k) code.

- 3.6 **Polar Codes:** Introduced by E. Arikan in 2009, polar codes are linear codes which provably achieve the capacity of symmetric binary DMC (B-DMC). Channel polarisation is an operation which produces N channels from N independent copies of a B-DMC, such that the new parallel channels are polarised in the sense that their mutual information is either close to 0 (completely noisy channels) or close to 1 (perfectly noiseless channels). The polar encoder chooses a set of NR rows of the matrix \mathbf{G}_n to form a $NR \times N$ matrix, which is then used as the generator matrix in the encoding procedure. Write a program to implement polar encoder and decoder. Determine its performance over BSC with different values of p .

REFERENCES FOR FURTHER READING

1. Roth, R. M., *Introduction to Coding Theory*, Cambridge University Press, 2006.
2. Lin, S. and Costello, D. J., *Error Control Coding* (2nd Edition), Prentice-Hall, 2004.
3. Blahut, R. E., *Algebraic Codes for Data Transmission*, Cambridge University Press, 2002.

Cyclic Codes

We arrive at truth, not by reason only, but also by the heart.

Blaise Pascal (1623–1662)

Learning objectives

After studying this chapter the students will be able to:

- LO 1** Recall the knowledge of polynomials, which will come handy in studying cyclic codes.
- LO 2** Know how to generate cyclic codes and to represent cyclic codes using generator polynomials and generator matrices.
- LO 3** Describe some popular cyclic codes.
- LO 4** Explain how to implement cyclic codes using circuits.

4.1 Introduction to Cyclic Codes

In the previous chapter, while dealing with Linear Block Codes (LBCs), certain linearity constraints were imposed on the structure of the block codes. These structural properties help us to search for good LBCs which are fast and easy to encode and decode.

However, LBCs of very large block lengths can be prohibitive both in terms of memory and computational costs. Also, most of the good LBCs have been found by the sheer ingenuity of the inventors, and often carry the name of the inventor of the coding scheme. We have not studied any mechanism to *construct* or *design* the generator matrix or the parity check matrix.

In this chapter we shall explore a *subclass* of LBCs which has another constraint on the structure of the codes. The additional constraint may be caused by any cyclic shift of a codeword resulting in another valid codeword. This condition allows the simple implementation of these cyclic codes by using shift registers. Efficient circuit implementation is a selling feature of any error control code. Additionally, the cyclic codes can be efficiently designed. We shall also see how the theory of Galois Field can be used effectively to

...
This chapter comes with a video overview by the author. Scan here to know more or
[Visit <http://qrcode.flipick.com/index.php/601>](http://qrcode.flipick.com/index.php/601)



study, analyse and discover new cyclic codes. The Galois Field representation of cyclic codes leads to low-complexity encoding and decoding algorithms.

In this chapter

The first two sections involve a mathematical detour of Polynomials. We will refresh our knowledge of polynomials which will be used to represent cyclic codes efficiently. We will also revisit the division algorithm for polynomials, in LO 1.

Next, we will learn how to generate cyclic codes using operations on polynomials. Since cyclic codes form a sub-class of linear block codes, we will study the matrix description of cyclic codes. We will also look at quasi- and shortened-cyclic codes and find out how cyclic codes are effective in handling burst errors, in LO 2.

We will then study some famous cyclic codes, namely Fire codes, Golay codes and Cyclic Redundancy Check (CRC) codes, in LO 3.

Finally, we shall find out how to implement cyclic codes using circuit elements. We will also learn about an efficient decoder called the Meggitt decoder, in LO 4.

4.2 Polynomials

LO 1

Recall the knowledge of polynomials, which will come handy in studying cyclic codes.

Before we take a short detour to the world of polynomials, first, a quick definition:

Definition 4.1 A code \mathbf{C} is **cyclic** if

- (i) \mathbf{C} is a linear code, and,
- (ii) any cyclic shift of a codeword is also a codeword, i.e., if the codeword $a_0a_1\dots a_{n-1}$ is in \mathbf{C} then $a_{n-1}a_0\dots a_{n-2}$ is also in \mathbf{C} .

Example 4.1 The binary code $\mathbf{C}_1 = \{0000, 0101, 1010, 1111\}$ is a cyclic code.

However $\mathbf{C}_2 = \{0000, 0110, 1001, 1111\}$ is not a cyclic code, but is *equivalent* to the first code. Interchanging the third and the fourth components of \mathbf{C}_2 yield \mathbf{C}_1 .

The $(5, 2)$ linear code $\mathbf{C}_3 = \{00000, 01101, 11010, 10111\}$ is also not cyclic. The second codeword when cyclic-shifted to the left gives the third codeword. However, another left-shift does not yield a valid codeword.

Definition 4.2 A **polynomial** is a mathematical expression

$$f(x) = f_0 + f_1x + \dots + f_mx^m \quad (4.1)$$

where the symbol x is called the indeterminate and the coefficients f_0, f_1, \dots, f_m are the elements of $GF(q)$. The coefficient f_m is called the leading coefficient. If $f_m \neq 0$, then m is called the **degree** of the polynomial, and is denoted by $\deg f(x)$.

A polynomial is a convenient way to represent a vector. For example, the polynomial $f(x) = f_0 + f_1x + f_2x^2 + \dots + f_mx^m$ corresponds to the vector $[f_0, f_1, f_2 \dots f_m]$. We observe a one-to-one relation between a polynomial and a vector (word). Thus, a codeword, c , of length n can be expressed as a polynomial $c(x)$ as following:

$$c = [c_0 \ c_1 \ c_2 \ \dots \ c_{n-1}] \leftrightarrow c(x) = c_0 + c_1x + c_2x^2 + \dots + c_{n-1}x^{n-1}$$

Hence, the binary word $[1 \ 0 \ 0 \ 1 \ 1] \leftrightarrow 1 + 0x + 0x^2 + 1x^3 + 1x^4 = 1 + x^3 + x^4$.

Definition 4.3 A polynomial is called **monic** if its leading coefficient is unity.

Example 4.2 $f(x) = 3 + 7x + x^3 + 5x^4 + x^6$ is a monic polynomial over $GF(8)$. The degree of this polynomial is 6. It is monic because the coefficient of x^6 is 1.

$g(x) = 1 + x + x^3 + x^4 + 3x^6$ is not a monic polynomial over $GF(8)$ because the coefficient of x^6 is not unity.

Polynomials play an important role in the study of cyclic codes, the subject of this chapter. Let $F[x]$ be the set of polynomials in x with coefficients in $GF(q)$. Different polynomials in $F[x]$ can be added, subtracted and multiplied in the usual manner. $F[x]$ is an example of an algebraic structure called a *ring*. A ring satisfies the first seven of the eight axioms that define a field (see Section 3.2 of Chapter 3). $F[x]$ is not a field because polynomials of degree greater than zero do not have a multiplicative inverse. It can be seen that if $f(x), g(x) \in F[x]$, then $\deg(f(x)g(x)) = \deg f(x) + \deg g(x)$. However, $\deg(f(x) + g(x))$ is not necessarily $\max\{\deg f(x), \deg g(x)\}$. For example, let us consider the two polynomials, $f(x)$ and $g(x)$, over $GF(2)$ such that $f(x) = 1 + x^2$ and $g(x) = 1 + x + x^2$. Then, $\deg(f(x) + g(x)) = \deg(x) = 1$. This is because, in $GF(2)$, $1 + 1 = 0$, and $x^2 + x^2 = (1 + 1)x^2 = 0$.

Example 4.3 Consider the polynomials $f(x) = 2 + x + x^2 + 2x^4$ and $g(x) = 1 + 2x^2 + 2x^4 + x^5$ over $GF(3)$. Then,

$$\begin{aligned} f(x) + g(x) &= (2 + 1) + x + (1 + 2)x^2 + (2 + 2)x^4 + x^5 = x + x^4 + x^5. \\ f(x) \cdot g(x) &= (2 + x + x^2 + 2x^4)(1 + 2x^2 + 2x^4 + x^5) \\ &= 2 + x + (1 + 2.2)x^2 + 2x^3 + (2 + 2 + 2.2)x^4 + (2 + 2)x^5 \\ &\quad + (1 + 2 + 1)x^6 + x^7 + 2.2x^8 + 2x^9 \\ &= 2 + x + (1 + 1)x^2 + 2x^3 + (2 + 2 + 1)x^4 + (2 + 2)x^5 \\ &\quad + (1 + 2 + 1)x^6 + x^7 + x^8 + 2x^9 \\ &= 2 + x + 2x^2 + 2x^3 + 2x^4 + x^5 + x^6 + x^7 + x^8 + 2x^9 \end{aligned}$$

Note that the addition and multiplication of the coefficients have been carried out over $GF(3)$.

Example 4.4 Consider the polynomial $f(x) = 1 + x$ over $GF(2)$.

$$(f(x))^2 = 1 + (1 + 1)x + x^2 = 1 + x^2$$

Again consider $f(x) = 1 + x$ over $GF(3)$.

$$(f(x))^2 = 1 + (1 + 1)x + x^2 = 1 + 2x + x^2$$

4.3 The Division Algorithm for Polynomials

The division algorithm states that, for every pair of polynomial $a(x)$ and $b(x) \neq 0$ in $F[x]$, there exists a unique pair of polynomials $q(x)$, the quotient, and $r(x)$, the remainder, such that $a(x) = q(x)b(x) + r(x)$, where $\deg r(x) < \deg b(x)$. This is also called the **Euclidean division algorithm**.

The remainder is sometimes also called the **residue**, and is denoted by $R_{b(x)}[a(x)] = r(x)$.

Two important properties of residues are as follows:

$$(i) R_{f(x)}[a(x) + b(x)] = R_{f(x)}[a(x)] + R_{f(x)}[b(x)], \text{ and} \quad (4.2)$$

$$(ii) R_{f(x)}[a(x) \cdot b(x)] = R_{f(x)}\{R_{f(x)}[a(x)] \cdot R_{f(x)}[b(x)]\} \quad (4.3)$$

Where $a(x)$, $b(x)$ and $f(x)$ are polynomials over $GF(q)$.

Example 4.5 Let the polynomials, $a(x) = x^3 + x + 1$ and $b(x) = x^2 + x + 1$ be defined over $GF(2)$.

We can carry out the long division of $a(x)$ by $b(x)$ as follows:

$$\begin{array}{r} x+1 \longleftarrow q(x) \\ \overline{x^3+x+1} \qquad \qquad \qquad \longleftarrow a(x) \\ x^3+x^2+x \\ \hline x^2+1 \\ x^2+x+1 \\ \hline x \qquad \longleftarrow r(x) \end{array}$$

Thus, $a(x) = (x+1)b(x) + x$. Hence, we may write $a(x) = q(x)b(x) + r(x)$, where $q(x) = x+1$ and $r(x) = x$. Note that $\deg r(x) < \deg b(x)$.

Definition 4.4 Let $f(x)$ be a fixed polynomial in $F[x]$. Two polynomials, $g(x)$ and $h(x)$ in $F[x]$ are said to be **congruent modulo $f(x)$** , depicted by

$$g(x) \equiv h(x) \pmod{f(x)}$$

if $g(x) - h(x)$ is divisible by $f(x)$.

Example 4.6 Let the polynomials $g(x) = x^9 + x^2 + 1$, $h(x) = x^5 + x^2 + 1$ and $f(x) = x^4 + 1$ be defined over $GF(2)$. Since $g(x) - h(x) = x^5f(x)$, we can write $g(x) \equiv h(x) \pmod{f(x)}$.

Next, let us denote $F[x]/f(x)$ as the *set of polynomials* in $F[x]$ of degree less than $\deg f(x)$, with addition and multiplication carried out modulo $f(x)$ as follows:

- (i) If $a(x)$ and $b(x)$ belong to $F[x]/f(x)$, then the sum $a(x) + b(x)$ in $F[x]/f(x)$ is the same as in $F[x]$. This is because $\deg a(x) < \deg f(x)$, $\deg b(x) < \deg f(x)$ and therefore $\deg (a(x) + b(x)) < \deg f(x)$.
- (ii) The product $a(x)b(x)$ is the unique polynomial of degree less than $\deg f(x)$ to which $a(x)b(x)$ (multiplication being carried out in $F[x]$) is congruent modulo $f(x)$.

$F[x]/f(x)$ is called the *ring of polynomials* (over $F[x]$) modulo $f(x)$. As mentioned earlier in Definition 3.9, a ring satisfies the first seven of the eight axioms that define a field. A ring in which every element also has a multiplicative inverse forms a field.

Example 4.7 Consider the product $(x + 1)^2$ in $F[x]/(x^2 + x + 1)$ defined over $GF(2)$.

$$(x + 1)^2 = x^2 + x + x + 1 = x^2 + 1 = x \pmod{x^2 + x + 1}$$

The product $(x + 1)^2$ in $F[x]/(x^2 + 1)$ defined over $GF(2)$ can be expressed as

$$(x + 1)^2 = x^2 + x + x + 1 = x^2 + 1 = 0 \pmod{x^2 + x + 1}$$

The product $(x + 1)^2$ in $F[x]/(x^2 + x + 1)$ defined over $GF(3)$ can be expressed as

$$(x + 1)^2 = x^2 + x + x + 1 = x^2 + 2x + 1 = x \pmod{x^2 + x + 1}.$$

If $f(x)$ has degree n , then the ring $F[x]/f(x)$ over $GF(q)$ consists of polynomials of degree $\leq n - 1$. The size of ring will be q^n because each of the n coefficients of the polynomials can be one of the q elements in $GF(q)$.

Example 4.8 Consider the ring $F[x]/(x^2 + x + 1)$ defined over $GF(2)$. This ring will have

polynomials with highest degree = 1. This ring contains $q^n = 2^2 = 4$ elements (each element is a polynomial). The elements of the ring will be 0, 1, x and $x + 1$. The addition and multiplication tables can be written as follows.

Table 4.1 Addition and multiplication tables for the field $F[x]/(x^2 + x + 1)$

+	0	1	x	$x + 1$
0	0	1	x	$x + 1$
1	1	0	$x + 1$	x
x	x	$x + 1$	0	1
$x + 1$	$x + 1$	x	1	0

·	0	1	x	$x + 1$
0	0	0	0	0
1	0	1	x	$x + 1$
x	0	x	$x + 1$	1
$x + 1$	0	$x + 1$	1	x

Next, consider $F[x]/(x^2 + 1)$ defined over $GF(2)$. The elements of the ring will be 0, 1, x and $x + 1$. The addition and multiplication tables can be written as follows.

Table 4.2 Addition and multiplication tables for the ring $F[x]/(x^2 + 1)$

+	0	1	x	$x + 1$
0	0	1	x	$x + 1$
1	1	0	$x + 1$	x
x	x	$x + 1$	0	1
$x + 1$	$x + 1$	x	1	0

·	0	1	x	$x + 1$
0	0	0	0	0
1	0	1	x	$x + 1$
x	0	x	1	$x + 1$
$x + 1$	0	$x + 1$	$x + 1$	0

It is interesting to note that $F[x]/(x^2 + x + 1)$ is actually a field as the multiplicative inverse for all the non-zero elements also exist. On the other hand, $F[x]/(x^2 + 1)$ is *not* a field because the multiplicative inverse of element $x + 1$ does not exist.

It is worthwhile exploring the properties of $f(x)$ which makes $F[x]/f(x)$ a field. As we shall shortly find out, the polynomial $f(x)$ must be *irreducible (non-factorisable)*.

Definition 4.5 A polynomial $f(x)$ in $F[x]$ is said to be **reducible** if $f(x) = a(x)b(x)$, where $a(x), b(x)$ are elements of $F(x)$ and $\deg a(x)$ and $\deg b(x)$ are both smaller than $\deg f(x)$. If $f(x)$ is not reducible, it is called **irreducible**. A monic irreducible polynomial of degree at least one is called a **prime polynomial**.

It is helpful to compare a reducible polynomial with a positive integer that can be factorised into a product of prime numbers. Any monic polynomial in $f(x)$ can be factorised uniquely into a product of irreducible monic polynomials (prime polynomials). One way to verify a prime polynomial is by trial and error, testing for all possible factorisations. This would require a computer search. Prime polynomials of every degree exist over every Galois field.

Theorem 4.1

- (i) A polynomial $f(x)$ has a linear factor $(x - a)$ if and only if $f(a) = 0$ where a is a field element.
- (ii) A polynomial $f(x)$ in $F[x]$ of degree 2 or 3 over $GF(q)$ is irreducible if, and only if $f(a) \neq 0$ for all a in $GF(q)$.
- (iii) Over any field, $x^n - 1 = (x - 1)(x^{n-1} + x^{n-2} + \dots + x + 1)$. The second factor may be further reducible.

Proof:

- (i) If $f(x) = (x - a)g(x)$, then obviously $f(a) = 0$. On the other hand if $f(a) = 0$, by division algorithm, $f(x) = q(x)(x - a) + r(x)$, where $\deg r(x) < \deg (x - a) = 1$. This implies that $r(x)$ is a constant. But, since $f(a) = 0$, $r(x)$ must be zero, and therefore, $f(x) = q(x)(x - a)$.
- (ii) A polynomial of degree 2 or 3 over $GF(q)$ will be reducible if and only if it has at least one linear factor. The result (ii) then directly follows from (i). This result does not necessarily hold for polynomials of degree more than 3. This is because it might be possible to factorise a polynomial of degree 4 or higher into a product of polynomials, none of which are linear, i.e., of the type $(x - a)$.
- (iii) From (i), $(x - 1)$ is a factor of $(x^n - 1)$. By carrying out long division of $(x^n - 1)$ by $(x - 1)$ we obtain $(x^{n-1} + x^{n-2} + \dots + x + 1)$.

Example 4.9 Consider $f(x) = x^3 - 1$ over $GF(2)$. Using (iii) of Theorem 4.1 we can write

$x^3 - 1 = (x - 1)(x^2 + x + 1)$. This factorisation is true over any field. Now, let's try to factorise the second term, $p(x) = (x^2 + x + 1)$.

$$\begin{aligned} p(0) &= 0 + 0 + 1 = 1, \text{ over } GF(2) \\ p(1) &= 1 + 1 + 1 = 1, \text{ over } GF(2) \end{aligned}$$

Therefore, $p(x)$ cannot be factorised further (from (ii)).

Thus, over $GF(2)$, $x^3 - 1 = (x - 1)(x^2 + x + 1)$.

Next, consider $f(x) = x^3 - 1$ over $GF(3)$.

$$\begin{aligned} x^3 - 1 &= (x - 1)(x^2 + x + 1). \text{ Again, let } p(x) = (x^2 + x + 1) \\ p(0) &= 0 + 0 + 1 = 1, \text{ over } GF(3) \\ p(1) &= 1 + 1 + 1 = 0, \text{ over } GF(3) \\ p(2) &= 2 \cdot 2 + 2 + 1 = 1 + 2 + 1 = 1, \text{ over } GF(3) \end{aligned}$$

Since, $p(1) = 0$, from (i) we have $(x - 1)$ as a factor of $p(x)$.

Thus, over $GF(3)$, $x^3 - 1 = (x - 1)(x - 1)(x - 1)$.

Theorem 4.2 The ring $F[x]/f(x)$ is a field if and only if $f(x)$ is a prime polynomial in $F[x]$.

Proof: To prove that a ring is a field we must show that every non-zero element of the ring has a multiplicative inverse. Let $s(x)$ be a non-zero element of the ring.

We have, $\deg s(x) < \deg f(x)$, because $s(x)$ is contained in the ring $F[x]/f(x)$.

It can be shown that the greatest common divisor (GCD) of two polynomials $f(x)$ and $s(x)$ can be expressed as

$$\text{GCD}(f(x), s(x)) = a(x)f(x) + b(x)s(x)$$

where $a(x)$ and $b(x)$ are polynomials over $GF(q)$. Since $f(x)$ is irreducible in $F[x]$, we have

$$\text{GCD}(f(x), s(x)) = 1 = a(x)f(x) + b(x)s(x).$$

$$\begin{aligned}
\text{Now, } 1 &= R_{f(x)}[1] = R_{f(x)}[a(x)f(x) + b(x)s(x)] \\
&= R_{f(x)}[a(x)f(x)] + R_{f(x)}[b(x)s(x)] \quad (\text{property (i) of residues}) \\
&= 0 + R_{f(x)}[b(x)s(x)] \\
&= R_{f(x)}\{R_{f(x)}[b(x)] \cdot R_{f(x)}[s(x)]\} \quad (\text{property (ii) of residues}) \\
&= R_{f(x)}\{R_{f(x)}[b(x)] \cdot s(x)\}
\end{aligned}$$

Hence, $R_{f(x)}[b(x)]$ is the multiplicative inverse of $s(x)$.

Next, let us prove the *only if* part of the theorem. Let us suppose $f(x)$ has a degree of at least 2, and is *not* a prime polynomial (a polynomial of degree one is always irreducible). Therefore, we can write $f(x) = r(x)s(x)$ for some polynomials $r(x)$ and $s(x)$ with a degree of at least one. If the ring $F[x]/f(x)$ is indeed a field, then a multiplicative index of $r(x)$, $r^{-1}(x)$ exists, since all polynomials in the field must have their corresponding multiplicative inverses. Hence,

$s(x) = R_{f(x)}\{s(x)\} = R_{f(x)}\{r(x)r^{-1}(x)s(x)\} = R_{f(x)}\{r^{-1}(x)r(x)s(x)\} = R_{f(x)}\{r^{-1}(x)f(x)\} = 0$. However, we had assumed $s(x) \neq 0$. Thus, there is a contradiction which implies that the ring is not a field.

Note that a prime polynomial is both monic and irreducible. In the above theorem, it is sufficient to have $f(x)$ as irreducible in order to obtain a field. The theorem could as well be stated as: ‘The ring $F[x]/f(x)$ is a field if, and only if, $f(x)$ is irreducible in $F[x]$ ’.

So, we now have an elegant mechanism of generating Galois Fields! If we can identify a prime polynomial of degree n over $GF(q)$, we can construct a Galois Field with q^n elements. Such a field will have polynomials as the elements of the field. These polynomials will be defined over $GF(q)$ and consist of all polynomials of degree less than n . It can be seen that there will be q^n such polynomials, which form the elements of the field.

Example 4.10 Consider the polynomial $p(x) = x^3 + x + 1$ over $GF(2)$. Since, $p(0) \neq 0$ and $p(1) \neq 0$,

the polynomial is irreducible in $GF(2)$. Since it is also monic, $p(x)$ is a prime polynomial. Here we have $n = 3$, so we can use $p(x)$ to construct a field with $2^3 = 8$ elements, i.e., $GF(8)$. The elements of this field will be $0, 1, x, x + 1, x^2, x^2 + 1, x^2 + x, x^2 + x + 1$, which are all possible polynomials of degree less than $n = 3$. It is easy to construct the addition and multiplication tables for this field (exercise).

Having developed the necessary mathematical tools, we now resume our study of cyclic codes. We now fix $f(x) = x^n - 1$ for the remainder of the chapter. We also denote $F[x]/f(x)$ by R_n . Before we proceed, we make the following observations:

- (i) $x^n = 1 \pmod{x^n - 1}$. Hence, any polynomial, modulo $x^n - 1$, can be reduced simply by replacing x^n by 1, x^{n+1} by x and so on.
- (ii) A codeword can *uniquely* be represented by a polynomial. A codeword consists of a sequence of elements. We can use a polynomial to represent the locations and the values of all the elements in the codeword. For example, the codeword $c_0c_1c_2\dots c_{n-1}$ can be represented by the polynomial $c(x) = c_0 + c_1x + c_2x^2 + \dots + c_{n-1}x^{n-1}$. Thus,

$$c = c_0c_1c_2\dots c_{n-1} \leftrightarrow c_0 + c_1x + c_2x^2 + \dots + c_{n-1}x^{n-1} = c(x)$$

As another example, the codeword over $GF(8)$, $c = 207735$ can be represented by the polynomial $c(x) = 2 + 7x^2 + 7x^3 + 3x^4 + 5x^5$.

- (iii) Multiplying any polynomial by x corresponds to a single cyclic right-shift of the codeword elements. More explicitly, in R_n , by multiplying $c(x)$ by x we get

$$x \cdot c(x) = c_0x + c_1x^2 + c_2x^3 + \dots + c_{n-1}x^n = c_{n-1} + c_0x + c_1x^2 + c_2x^3 + \dots + c_{n-2}x^{n-1}$$



Theorem 4.3 A code \mathbf{C} in R_n is a cyclic code if and only if \mathbf{C} satisfies the following conditions:

$$(i) \quad a(x), b(x) \in \mathbf{C} \Rightarrow a(x) + b(x) \in \mathbf{C} \quad (4.4)$$

$$(ii) \quad a(x) \in \mathbf{C} \text{ and } r(x) \in R_n \Rightarrow a(x)r(x) \in \mathbf{C} \quad (4.5)$$

Proof:

- (i) Suppose \mathbf{C} is a cyclic code in R_n . Since cyclic codes are a subset of LBCs, the first condition holds.
- (ii) Let $r(x) = r_0 + r_1x + r_2x^2 + \dots + r_nx^n$. Multiplication by x corresponds to a cyclic right-shift. But, by definition, the cyclic shift of a cyclic codeword is also a valid codeword. That is, $x.a(x) \in \mathbf{C}$, $x.(xa(x)) \in \mathbf{C}$, and so on. Hence,
 $r(x)a(x) = r_0a(x) + r_1xa(x) + r_2x^2a(x) + \dots + r_nx^na(x)$ is also in \mathbf{C} since each summand is also in \mathbf{C} .

Next, we need to prove the *only if* part of the theorem. Suppose (i) and (ii) hold. Take $r(x)$ to be a scalar. Then (i) implies that \mathbf{C} is linear. Take $r(x) = x$ in (ii), which shows that any cyclic shift also leads to a codeword. Hence (i) and (ii) imply that \mathbf{C} is a cyclic code.

In the next section, we shall use the mathematical tools developed so far to construct cyclic codes.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 1:

1. Give two examples of monic polynomials over $GF(3)$. S
2. Consider a ring of 6 elements. Simplify the following, assuming that the coefficients follow the addition and multiplication rules for the ring. S
 - (i) $(2x + 1) + (4x + 5) + (3x + 3)$
 - (ii) $(2x + 1)^2$
 - (iii) $(4x + 3)(x + 5)$
3. Factorise $x^{15} - 1$ over $GF(2)$. S
4. Divide $x^7 + 1$ by $x^3 + x + 1$ over $GF(2)$. S
5. Divide $x^{12} + x^7 + x^4 + x^3 + 1$ by $x^3 + x^2 + 1$ over $GF(2)$ and give the quotient and remainder. M

If you have successfully solved the above problems,
you have mastered LO 1. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/596>



Levels of Difficulty

- Simple:** Level 1 and Level 2 Category
- Medium:** Level 3 and Level 4 Category
- Difficult:** Level 5 and Level 6 Category

4.4 A Method for Generating Cyclic Codes

The following steps can be used to generate a cyclic code:

- Take a polynomial $f(x)$ in R_n .
- Obtain a set of polynomials by multiplying $f(x)$ by all possible polynomials in R_n .
- The set of polynomials obtained above corresponds to the set of codewords belonging to a cyclic code. The block length of the code is n .

LO 2



Know how to generate cyclic codes and to represent cyclic codes using generator polynomials and generator matrices.

Example 4.11 Consider the polynomial $f(x) = 1 + x^2$ in R_3 defined over $GF(2)$. In general, a polynomial in R_3 ($= F[x]/(x^3 - 1)$) can be represented as $r(x) = r_0 + r_1x + r_2x^2$, where the coefficients can take the values 0 or 1 (since defined over $GF(2)$). Thus, there can be a total of $2 \times 2 \times 2 = 8$ polynomials in R_3 defined over $GF(2)$, which are 0, 1, x , x^2 , $1 + x$, $1 + x^2$, $x + x^2$, $1 + x + x^2$. To generate the cyclic code, we multiply $f(x)$ with these 8 possible elements of R_3 and then reduce the results modulo $(x^3 - 1)$:

$$\begin{aligned}(1 + x^2) \cdot 0 &= 0, (1 + x^2) \cdot 1 = (1 + x^2), (1 + x^2) \cdot x = 1 + x, (1 + x^2) \cdot x^2 = x + x^2 \\ (1 + x^2) \cdot (1 + x) &= x + x^2, (1 + x^2) \cdot (1 + x^2) = 1 + x, (1 + x^2) \cdot (x + x^2) = (1 + x^2), \\ (1 + x^2) \cdot (1 + x + x^2) &= 0\end{aligned}$$

Thus there are only four distinct codewords: $\{0, 1 + x, 1 + x^2, x + x^2\}$ which correspond to $\{000, 011, 101, 110\}$.

From the above example it appears that we can have some sort of a **generator polynomial** which can be used to construct the cyclic code.

Theorem 4.4 Let C be a (n, k) non-zero cyclic code in R_n . Then

- there exists a unique monic polynomial $g(x)$ of the smallest degree in C ,
- the cyclic code C consists of all multiples of the generator polynomial $g(x)$ by polynomials of degree $k - 1$ or less,
- $g(x)$ is a factor of $x^n - 1$.

Proof:

- Suppose both $g(x)$ and $h(x)$ are monic polynomials in C of smallest degree. Then $g(x) - h(x)$ is also in C , and has a smaller degree. If $g(x) \neq h(x)$, then a suitable scalar multiplier of $g(x) - h(x)$ is monic, and is in C , and is of smaller degree than $g(x)$. This gives a contradiction.
- Let $a(x) \in C$. Then, by division algorithm, $a(x) = q(x)g(x) + r(x)$, where $\deg r(x) < \deg g(x)$. But $r(x) = a(x) - q(x)g(x) \in C$ because both words on the right hand side of the equation are codewords. However, the degree of $g(x)$ must be the minimum among all codewords. This can only be possible if $r(x) = 0$ and $a(x) = q(x)g(x)$. Thus, a codeword is obtained by multiplying the generator polynomial

$g(x)$ with the polynomial $q(x)$. For a code defined over $GF(q)$, here are q^k distinct codewords possible. These codewords correspond to multiplying $g(x)$ with the q^k distinct polynomials, $q(x)$, where $\deg q(x) \leq (k - 1)$.

- (iii) By division algorithm, $x^n - 1 = q(x)g(x) + r(x)$, where $\deg r(x) < \deg g(x)$ or $r(x) = \{(x^n - 1) - q(x)g(x)\}$ modulo $(x^n - 1) = -q(x)g(x)$. But $-q(x)g(x) \in \mathbf{C}$ because we are multiplying the generator polynomial by another polynomial, $-q(x)$. Thus, we have a codeword $r(x)$ whose degree is less than that of $g(x)$. This violates the minimality of the degree of $g(x)$, unless $r(x) = 0$, which implies $x^n - 1 = q(x)g(x)$, i.e., $g(x)$ is a factor of $x^n - 1$.

The last part of the theorem gives us the recipe to obtain the generator polynomial for a cyclic code. All we have to do is to factorise $x^n - 1$ into irreducible, monic polynomials. We can also find all the possible cyclic codewords of blocklength n simply by factorising $x^n - 1$.

 **Note 1:** A cyclic code \mathbf{C} may contain polynomials other than the generator polynomial which *also* generates \mathbf{C} . But the polynomial with the minimum degree is called the generator polynomial.

Note 2: The degree of $g(x)$ is $n - k$ (this will be shown later).

Example 4.12 To find all the binary codes of blocklength 3, we first factorise $x^3 - 1$. Note that for $GF(2)$, $1 = -1$, since $1 + 1 = 0$. Hence, $x^3 - 1 = x^3 + 1 = (x + 1)(x^2 + x + 1)$. Thus, we can construct Table 4.3.

Table 4.3 All possible binary cyclic codes of block length 3

Generator Polynomial	Code (polynomial)	Code (binary)
1	$\{R_3\}$	$\{000, 001, 010, 011, 100, 101, 110, 111\}$
$(x + 1)$	$\{0, x + 1, x^2 + x, x^2 + 1\}$	$\{000, 011, 110, 101\}$
$(x^2 + x + 1)$	$\{0, x^2 + x + 1\}$	$\{000, 111\}$
$(x^3 + 1) = 0$	$\{0\}$	$\{000\}$

A simple encoding rule to generate the codewords from the generator polynomial is

$$c(x) = i(x)g(x) \quad (4.6)$$

where $i(x)$ is the information polynomial, $c(x)$ is the codeword polynomial and $g(x)$ is the generator polynomial. We have seen already that there is a one-to-one correspondence between a word (vector) and a polynomial. The error vector can also be represented as the error polynomial, $e(x)$. Thus, the received word at the receiver, after passing through a noisy channel can be expressed as

$$v(x) = c(x) + e(x) \quad (4.7)$$

We define the **Syndrome Polynomial**, $s(x)$ as the remainder of $v(x)$ under division by $g(x)$, i.e.,

$$s(x) = R_{g(x)}[v(x)] = R_{g(x)}[c(x) + e(x)] = R_{g(x)}[c(x)] + R_{g(x)}[e(x)] = R_{g(x)}[e(x)] \quad (4.8)$$

because $R_{g(x)}[c(x)] = 0$ from (4.6).

Example 4.13 Consider the generator polynomial $g(x) = x^3 + x + 1$ for a binary cyclic code (i.e., over $GF(2)$) of block length $n = 7$. The codewords and the corresponding codeword polynomials are listed in Table 4.4. It is clear from the table that the minimum weight of this code is 3. This is, in fact, the $(7, 4)$ Hamming code. Observe how every codeword is a cyclic shift of some other codeword.

Table 4.4 Binary cyclic code constructed using $g(x) = x^3 + x + 1$

c	$c(x)$
0000000	0
0001011	$x^3 + x + 1$
0010110	$x^4 + x^2 + x$
0011101	$x^4 + x^3 + x^2 + 1$
0100111	$x^5 + x^2 + x + 1$
0101100	$x^5 + x^3 + x^2$
0110001	$x^5 + x^4 + 1$
0111010	$x^5 + x^4 + x^3 + 1$
1000101	$x^6 + x^2 + 1$
1001110	$x^6 + x^3 + x^2 + x$
1010011	$x^6 + x^4 + x + 1$
1011000	$x^6 + x^4 + x^3$
1100010	$x^6 + x^5 + x$
1101001	$x^6 + x^5 + x^3 + 1$
1110100	$x^6 + x^5 + x^4 + x^2$
1111111	$x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$

Example 4.14 Consider the generator polynomial $g(x) = x^2 + 1$ for ternary cyclic codes (i.e., over $GF(3)$) of block length $n = 4$. Here we are dealing with cyclic codes, therefore, the highest power of $g(x)$ is $n - k$. Since $n = 4$, k must be 2. So, we are going to construct a $(4, 2)$ cyclic ternary code. There will be a total of $q^k = 3^2 = 9$ code words. The information polynomials and the corresponding code word polynomials are listed in Table 4.5.

Table 4.5 Ternary cyclic code constructed using $g(x) = x^2 + 1$

I	$i(x)$	$c(x) = i(x)g(x)$	C
00	0	0	0000
01	1	$x^2 + 1$	0101
02	2	$2x^2 + 2$	0202
10	x	$x^3 + x$	1010
11	$x + 1$	$x^3 + x^2 + x + 1$	1111
12	$x + 2$	$x^3 + 2x^2 + x + 2$	1212
20	$2x$	$2x^3 + 2x$	2020
21	$2x + 1$	$2x^3 + x^2 + 2x + 1$	2121
22	$2x + 2$	$2x^3 + 2x^2 + 2x + 2$	2222

It can be seen that the cyclic shift of any code word results in another valid code word. By observing the codewords we find that the minimum distance of this code is 2 (there are four non-zero codewords with the minimum Hamming weight = 2). Therefore, this code is capable of detecting one error and correcting zero errors.

Observing the fact that the codeword polynomial is divisible by the generator polynomial, we can *detect* more number of errors than suggested by the minimum distance of the code. Since we are dealing with cyclic codes that are a subset of LBCs, we can use the all-zero codeword to illustrate this point without loss of generality.

Assume that $g(x) = x^2 + 1$ and the transmitted codeword is the all-zero codeword. Therefore, the received word is the error polynomial, i.e.,

$$v(x) = c(x) + e(x) = e(x) \quad (4.9)$$

At the receiver's end, an error will be detected if $g(x)$ fails to divide the received word $v(x) = e(x)$. Now, $g(x)$ has only two terms. So if the $e(x)$ has odd number of terms, i.e., if the number of errors are odd, it will be caught by the decoder! For example, if we try to divide $e(x) = x^3 + x + 1$ by $g(x)$, we will always get a remainder. In the case of the $(4, 2)$ cyclic code with $g(x) = x^2 + 1$, the $d^* = 2$, suggesting that it can detect $d^* - 1 = 1$ error. However, by this simple observation we find that it can detect any odd number of errors $\leq n$. In this case it can detect 1 error or 3 errors, but not 2 errors.

4.5 Matrix Description of Cyclic Codes

LO 2

Theorem 4.5 Suppose \mathbf{C} is a cyclic code with generator polynomial $g(x) = g_0 + g_1x + \dots + g_rx^r$ of degree r . Then the generator matrix of \mathbf{C} is given by

$$\mathbf{G} = \begin{bmatrix} g_0 & g_1 & \dots & g_r & 0 & 0 & 0 & \dots & 0 \\ 0 & g_0 & g_1 & \dots & g_r & 0 & 0 & \dots & 0 \\ 0 & 0 & g_0 & g_1 & \dots & g_r & 0 & \dots & 0 \\ \vdots & \vdots & & & & & & \vdots & \\ 0 & 0 & 0 & 0 & 0 & g_0 & g_1 & \dots & g_r \end{bmatrix} \quad \begin{array}{l} \uparrow \\ k = (n - r) \text{ rows} \\ \downarrow \\ \text{n columns} \end{array} \quad (4.10)$$

Proof: The $(n - r)$ rows of the matrix are obviously linearly independent because of the echelon form of the matrix. These $(n - r)$ rows represent the codewords $g(x), xg(x), x^2g(x), \dots, x^{n-r-1}g(x)$. Thus, the matrix can generate these codewords. Now, to prove that the matrix can generate *all* the possible codewords, we must show that every possible codeword can be represented as linear combinations of the codewords $g(x), xg(x), x^2g(x), \dots, x^{n-r-1}g(x)$.

We know that if $c(x)$ is a codeword, it can be represented as

$$c(x) = q(x) \cdot g(x)$$

for some polynomial $q(x)$. Since the degree of $c(x) < n$ (because the length of the codeword is n), it follows that the degree of $q(x) < n - r$. Hence,

$$q(x) \cdot g(x) = (q_0 + q_1x + \dots + q_{n-r-1}x^{n-r-1})g(x) = q_0g(x) + q_1xg(x) + \dots + q_{n-r-1}x^{n-r-1}g(x)$$

Thus, any codeword can be represented as a linear combination of $g(x), xg(x), x^2g(x), \dots, x^{n-r-1}g(x)$. This proves that the matrix \mathbf{G} is indeed the generator matrix.



We also know that the dimensions of the generator matrix is $k \times n$. Therefore, $r = n - k$, i.e., the degree of $g(x)$ is $n - k$. Note that the generator matrix has an interesting structure because it is constant along the diagonal. Such a matrix is called a **Toeplitz matrix**.

Example 4.15 To find the generator matrices of all ternary codes (i.e., codes over $GF(3)$) of blocklength $n = 4$, we first factorise $x^4 - 1$.

$$x^4 - 1 = (x - 1)(x^3 + x^2 + x + 1) = (x - 1)(x + 1)(x^2 + 1)$$

Thus all the divisors of $x^4 - 1$ are: 1, $(x - 1)$, $(x + 1)$, $(x^2 + 1)$, $(x - 1) \cdot (x + 1) = (x^2 - 1)$, $(x - 1) \cdot (x^2 + 1) = (x^3 - x^2 + x + 1)$, $(x + 1) \cdot (x^2 + 1) = (x^3 + x^2 + x + 1)$ and $(x^4 - 1)$. We know that all these factors of $x^4 - 1$ are capable of generating a cyclic code.

The corresponding generator matrices are listed in Table 4.6. Note that $-1 = 2$ for $GF(3)$.

It can be seen from Table 4.6 that none of the $(4, 2)$ ternary cyclic codes are single error correcting codes (since their minimum distance is less than 3). An interesting observation is that we do not have any

ternary $(4, 2)$ Hamming code that is cyclic! Remember, Hamming codes are single error correcting codes with $n = (q^r - 1)/(q - 1)$ and $k = (q^r - 1)/(q - 1) - r$, where r is an integer ≥ 2 . Therefore, a $(4, 2)$ ternary Hamming code exists, but it is not a cyclic code.

Table 4.6 Cyclic codes of blocklength $n = 4$ over $GF(3)$.

$g(x)$	(n, k)	d^*	\mathbf{G}
1	$(4, 4)$	1	$[I_4]$
$(x - 1)$	$(4, 3)$	2	$\begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$
$(x + 1)$	$(4, 3)$	2	$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$
$(x^2 + 1)$	$(4, 2)$	2	$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$
$(x^2 - 1)$	$(4, 2)$	2	$\begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$
$(x^3 - x^2 + x - 1)$	$(4, 1)$	4	$[-1 \ 1 \ -1 \ 1]$
$(x^3 + x^2 + x + 1)$	$(4, 1)$	4	$[1 \ 1 \ 1 \ 1]$
$(x^4 - 1)$	$(4, 1)$	0	$[0 \ 0 \ 0 \ 0]$

The next step is to explore if we can find a parity check polynomial corresponding to our generator polynomial, $g(x)$. We already know that $g(x)$ is a factor of $x^n - 1$. Hence we can write

$$x^n - 1 = h(x)g(x) \quad (4.11)$$

where $h(x)$ is some polynomial. Following can be concluded by simply observing the above equation:

- (i) Since $g(x)$ is monic, $h(x)$ has to be monic because the left hand side of the equation is also monic (the leading coefficient is unity).
- (ii) Since degree of $g(x)$ is $n - k$, the degree of $h(x)$ must be k .

Suppose C is a cyclic code in R_n with the generator polynomial $g(x)$. Recall that we are denoting $F[x]/f(x)$ by R_n , where $f(x) = x^n - 1$. In R_n , $h(x)g(x) = x^n - 1 = 0$. Then, any codeword belonging to C can be written as $c(x) = a(x)g(x)$, where the polynomial $a(x) \in R_n$. Therefore in R_n ,

$$c(x)h(x) = a(x)g(x)h(x) = c(x) \cdot 0 = 0 \quad (4.12)$$



Thus, $h(x)$ behaves like a **Parity Check Polynomial**. Any valid codeword when multiplied by the parity check polynomial yields the zero polynomial. This concept is parallel to that of the parity check matrix introduced in the previous chapter. Since we are still in the domain of LBCs, we go ahead and define the parity check matrix in relation to the parity check polynomial.

The parity check polynomial can be used to generate another cyclic code since it is a factor of $x^n - 1$ and is called the **dual code** of \mathbf{C} .

Example 4.16 Consider the generator polynomial $g(x) = x^3 + 1$ for the (9, 6) binary cyclic codes.

The parity check polynomial can be found by simply dividing $x^9 - 1$ by $g(x)$.

$$\text{Thus } h(x) = (x^9 - 1) / (x^3 + 1) = x^6 + x^3 + 1.$$

Therefore, the dual code of $g(x) = x^3 + 1$ is $h(x) = x^6 + x^3 + 1$.

Suppose C is a cyclic code with the parity check polynomial $h(x) = h_0 + h_1x + \dots + h_kx^k$, then the parity check matrix of C is given by

$$H = \begin{bmatrix} h_k & h_{k-1} & \dots & h_0 & 0 & 0 & 0 & \dots & 0 \\ 0 & h_k & h_{k-1} & \dots & h_0 & 0 & 0 & \dots & 0 \\ 0 & 0 & h_k & h_{k-1} & \dots & h_0 & 0 & \dots & 0 \\ \vdots & \vdots & & & & & & \vdots & \\ 0 & 0 & 0 & 0 & 0 & h_k & h_{k-1} & \dots & h_0 \end{bmatrix} \quad \begin{array}{c} \uparrow \\ (n-k) \text{ rows} \end{array} \quad \begin{array}{c} \downarrow \\ n \text{ columns} \end{array} \quad (4.13)$$

We observe that \mathbf{H} is a Toeplitz matrix. Recall that $c\mathbf{H}^T = \mathbf{0}$. Therefore, $i\mathbf{G}\mathbf{H}^T = \mathbf{0}$ for any information vector, i . Hence, $\mathbf{G}\mathbf{H}^T = \mathbf{0}$. We further have $s = v\mathbf{H}^T$ where s is the syndrome vector and v is the received word.

Example 4.17 For binary codes of blocklength, $n = 7$, we have

$$x^7 - 1 = (x - 1)(x^3 + x + 1)(x^3 + x^2 + 1)$$

Consider $g(x) = (x^3 + x + 1)$. Since $g(x)$ is a factor of $x^7 - 1$, there is a cyclic code that can be generated by it. The generator matrix corresponding to $g(x)$ is

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

The parity check polynomial $h(x)$ is $(x - 1)(x^3 + x^2 + 1) = (x^4 + x^2 + x + 1)$. And the corresponding parity check matrix is

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

The minimum distance of this code is 3, and this happens to be the (7, 4) Hamming code. Thus, the binary (7, 4) Hamming code is also a cyclic code with a generator polynomial $g(x) = (x^3 + x + 1)$.

Suppose we have to encode the information polynomial $i(x) = (x^2 + x + 1)$. This corresponds to the information word [1 1 1 0]. The corresponding codeword polynomial will be simply

$$c(x) = i(x)g(x) = (x^2 + x + 1)(x^3 + x + 1) = x^5 + x^4 + 1$$

This corresponds to the codeword [1 0 0 0 1 1 0].

We can verify the validity of the codeword polynomial by computing

$$\begin{aligned} s(x) &= c(x)h(x) = (x^5 + x^4 + 1)(x^4 + x^2 + x + 1) \bmod (x^7 - 1) \\ &= x^9 + x^8 + x^7 + x^2 + x + 1 \bmod (x^7 - 1) \\ &= (x^2 + x + 1)(x^7 - 1) \bmod (x^7 - 1) = 0 \end{aligned}$$

Note that in the above calculations, all ‘–’ can be replaced by ‘+’ since we are working with binary codes, i.e., over $GF(2)$.

4.6 Quasi-Cyclic Codes and Shortened Cyclic Codes

LO 2

In this section we will study codes that are closely related to cyclic codes.

Definition 4.6 An (n, k) **quasi-cyclic code** is a linear block code such that, for some m coprime with n , the polynomial $x^m c(x)$ calculated mod $x^n - 1$ is a codeword polynomial if $c(x)$ is a valid codeword polynomial.

Example 4.18 Consider the (12, 4) quasi-cyclic code given by the generator matrix

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

Here $m = 3$ because the cyclic shift of any row of \mathbf{G} by three places produces another row of the matrix.

Any cyclic code can be **Puncturing** to form a quasi-cyclic code by dropping every m^{th} symbol where m is a factor of n . This quasi-code will be a shortened code if none of the dropped symbols are check symbols. This means that all the dropped symbols are data symbols, which can be first set to zero.

Definition 4.7 An (n, k) linear code is called a proper **shortened cyclic code** if it is obtained by deleting m consecutive places from an $(n + m, k + m)$ cyclic code.

The unused symbols in a shortened code are always set to zero and are not transmitted. The receiver re-inserts the zeroes prior to decoding. Thus, even though the code is no longer cyclic, the same encoder and decoder can be used after the zeroes have been taken care of.

Example 4.19 Consider the (7, 4) Hamming code, which is also a cyclic code. We form a shortened (5, 3) code as follows. The two leading bits in the codeword are set to zero, followed by two information bits and the same three parity bits. The minimum distance of the shortened code is also 3, making it a single error correcting code.

4.7 Burst Error Correction

LO 2

DID YOU KNOW ? In many real life channels, errors are not random, but occur in bursts. For example, in a mobile communications channel, fading results in burst errors. When errors occur at a stretch, as opposed to random errors, we term them as **Burst Errors**.

Example 4.20 Let the transmitted sequence of bits, transmitted at 10 kb/s over a wireless channel, be

$$\mathbf{c} = 0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1$$

Suppose, after 0.5 ms of the start of transmission, the channel experiences a fade of duration 1 ms. During this time interval, the channel corrupts the transmitted bits. The error sequence can be written as

$$\mathbf{b} = 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0$$

This is an example of a burst error, where a portion of the transmitted sequence gets garbled due to the channel. Here the *length* of the burst is 10 bits, however, not all the ten locations are in error.



Definition 4.8 A **cyclic burst** of length t is a vector whose non-zero components are among t successive components, the first and last of which are non-zero.

If we are constructing codes for channels more prone to burst errors of length t (as opposed to an arbitrary pattern of t random errors) it might be possible to design more efficient codes. We can describe a burst error as

$$e(x) = x^i \cdot b(x) \quad (4.14)$$

where $e(x)$ is a polynomial of degree $\leq t - 1$ and $b(x)$ is the burst pattern. x^i marks the starting location of the burst pattern within the codeword being transmitted.

A code designed for correcting a burst of length t must have unique syndromes for every error pattern, i.e.,

$$s(x) = R_{g(x)}[e(x)] \quad (4.15)$$

is different for each polynomial representing a burst of length t .

Example 4.21 For a binary code of block length $n = 15$, consider the generator polynomial

$$g(x) = x^6 + x^3 + x^2 + x + 1$$

This code is capable of correcting bursts of length 3 or less. To prove this we must show that all the syndromes corresponding to the different burst errors are distinct. The different burst errors are

(i) Bursts of length 1

$$e(x) = x^i \text{ for } i = 0, 1, \dots, 14$$

(ii) Bursts of length 2

$$e(x) = x^i \cdot (1 + x) \text{ for } i = 0, 1, \dots, 13, \text{ and } e(x) = x^i \cdot (1 + x^2) \text{ for } i = 0, 1, \dots, 13$$

(iii) Bursts of length 3

$$e(x) = x^i \cdot (1 + x + x^2) \text{ for } i = 0, 1, \dots, 12$$

It can be shown that the syndrome of all these 56 ($15 + 14 + 14 + 13$) error patterns are distinct. A table can be made for each pattern and the corresponding syndrome which can be used for correcting a burst error of length 3 or less. It should be emphasised that the codes designed specifically for correcting burst errors are more efficient in terms of the code rate. The code being discussed here is a (15, 9) cyclic code with code rate $= k/n = 0.6$ and minimum distance $d^* = 3$. This code can correct only one random error (but up to three burst errors!). Note that correction of one random error amounts to correcting a burst error of length 1.

Similar to the Singleton Bound studied in the previous chapter, there is a bound for the minimum number of parity bits required for a burst-error correcting linear block code: ‘A linear block code that corrects all bursts of length t or less must have at least $2t$ parity symbols’. This is known as the **Rieger Bound**.

In the next three sections, we will study three different sub-classes of cyclic codes. Each sub-class has a specific objective.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 2:

- Find the generator polynomial of degree 5 for a cyclic code of block length 15 over $GF(2)$. S
- Find the generator matrix for a (7, 3) cyclic code over $GF(2)$. List all the codewords. S
- Find the parity check polynomial for the cyclic code, over $GF(2)$, with block length $n = 7$ and $g(x) = x^4 + x^3 + x^2 + 1$. Also, write the corresponding H matrix. S
- Determine all binary cyclic codes of block length $n = 9$. M
- Consider the generator polynomial $g(x) = (x^3 + x + 1)$ for a (7, 3) cyclic code over $GF(2)$. Let the received vector $r(x) = (x + x^2 + x^4 + x^5 + x^6)$. Calculate the syndrome polynomial for $r(x)$ and all its cyclic shifts. D

If you have successfully solved the above problems,
you have mastered LO 2. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/597>



4.8 Fire Codes

Definition 4.9 A Fire code is a cyclic burst error correcting code over $GF(q)$ with the generator polynomial

$$g(x) = (x^{2t-1} - 1)p(x) \quad (4.16)$$

where $p(x)$ is a prime polynomial over $GF(q)$ whose degree m is not smaller than t and $p(x)$ does not divide $x^{2t-1} - 1$. The block length of the Fire code is the smallest integer n such that $g(x)$ divides $x^n - 1$. A Fire code can correct all burst errors of length t or less.

LO 3

Describe some popular cyclic codes.

Example 4.22 Consider the Fire code with $t = m = 3$. A prime polynomial over $GF(2)$ of degree 3 is $p(x) = x^3 + x + 1$, which does not divide $(x^5 - 1)$. The generator polynomial of the Fire code will be

$$\begin{aligned} g(x) &= (x^5 - 1)p(x) = (x^5 - 1)(x^3 + x + 1) \\ &= x^8 + x^6 + x^5 - x^3 - x - 1 \\ &= x^8 + x^6 + x^5 + x^3 + x + 1 \end{aligned}$$

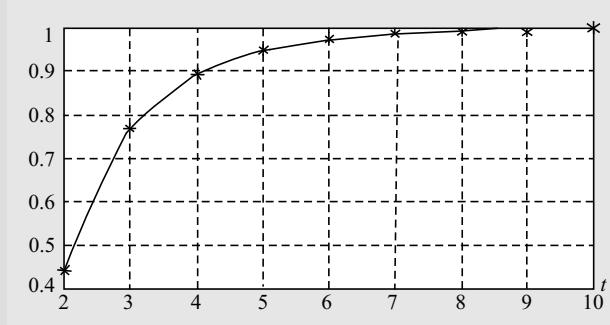


Fig. 4.1 Code rates for different fire codes.

The degree of $g(x) = n - k = 8$. The block length is the smallest integer n such that $g(x)$ divides $x^n - 1$. After trial and error we get $n = 35$. Thus, the parameters of the Fire code are $(35, 27)$ with $g(x) = x^8 + x^6 + x^5 + x^3 + x + 1$. This code can correct up to bursts of length 3. The code rate of this code is 0.77, and is more efficient than the code generated by $g(x) = x^6 + x^3 + x^2 + x + 1$ which has a code rate of only 0.6.

Fire codes become more efficient as we increase t . The code rates for binary Fire codes (with $m = t$) for different values of t are plotted in Fig. 4.1.

Example 4.23 The parameters of some of the popular binary Fire codes are given in the table.

(n, k)	t
(9, 4)	2
(35, 27)	3
(105, 94)	4
(279, 265)	5
(693, 676)	6
(1651, 1631)	7
(3825, 3802)	8

4.9 Golay Codes

LO 3

The Binary Golay Code

In the previous chapter, Section 3.9, we saw that a (23, 12) perfect code exists with $d^* = 7$. Recall that, for a perfect code,

$$M \left\{ \binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \dots + \binom{n}{t}(q-1)^t \right\} = q^n \quad (4.17)$$

which is satisfied for the values: $n = 23$, $k = 12$, $M = 2^k = 2^{12}$, $q = 2$ and $t = (d^* - 1)/2 = 3$. Thus (23, 12) perfect code is the binary Golay code. We shall now explore this perfect code as a cyclic code. We start with the factorisation of $(x^{23} - 1)$.

$$\begin{aligned} (x^{23} - 1) &= (x - 1)(x^{11} + x^{10} + x^6 + x^5 + x^4 + x^2 + 1)(x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1) \\ &= (x - 1)g_1(x)g_2(x) \end{aligned} \quad (4.18)$$

The degree of $g_1(x) = n - k = 11$, hence $k = 12$, which implies that there exists a (23, 12) cyclic code. In order to prove that it is a perfect code, we must show that the minimum distance of this (23, 12) cyclic code is 7. One way is to write out the parity check matrix, H , and show that no six columns are linearly dependent. Another way is to prove it analytically, which is a long and drawn-out proof. The easiest way is to write a computer program to list out all the 2^{12} codewords and find the minimum weight (on a fast computer it takes about 15 seconds!). The code rate is 0.52 and it is a triple error correcting code. However, the relatively small block length of this perfect code makes it impractical for most real life applications.

INDUSTRIAL RELEVANCE



The **Automatic Link Establishment** (ALE) protocol for the HF radio system used by NASA uses the extended (24, 12) Golay code.

The Ternary Golay Code

We next examine the ternary (11, 6) cyclic code, which is also the ternary Golay code. This code has a minimum distance = 5, and can be verified to be a perfect code. We begin by factorising $(x^{11} - 1)$ over $GF(3)$.

$$\begin{aligned} (x^{11} - 1) &= (x - 1)(x^5 + x^4 - x^3 + x^2 - 1)(x^5 - x^3 - x^2 - x - 1) \\ &= (x - 1)g_1(x)g_2(x) \end{aligned} \quad (4.19)$$

The degree of $g_1(x) = n - k = 5$, hence $k = 6$, which implies that there exists a (11, 6) cyclic code. In order to prove that it is a perfect code, we must show that the minimum distance of this (11, 6) cyclic code is 5. Again, we resort to an exhaustive computer search and find that the minimum distance is indeed 5.

It can be shown that $(x^p - 1)$ has a factorisation of the form $(x - 1)g_1(x)g_2(x)$ over $GF(2)$, whenever p is a prime number of the form $8m \pm 1$ (m is a positive integer). In such cases, $g_1(x)$ and $g_2(x)$ generate equivalent codes. If the minimum distance of the code generated by $g_1(x)$ is odd, then it satisfies the **square root bound**

$$d^* \geq \sqrt{n} \quad (4.20)$$

Note that n denotes the block length.

4.10 Cyclic Redundancy Check (CRC) Codes

LO 3

One of the common error detecting codes is the CRC codes. For a k -bit block of bits the (n, k) CRC encoder generates $(n - k)$ bit long frame check sequence (FCS). Let us define the following.

T = n -bit frame to be transmitted

D = k -bit message block (information bits)

F = $(n - k)$ bit FCS, the last $(n - k)$ bits of T

P = the predetermined divisor, a pattern of $(n - k + 1)$ bits.

Since the pre-determined divisor, P , should be able to divide the codeword T . Thus, T/P has no remainder. Now, D is the k -bit message block. Therefore, $2^{n-k}D$ amounts to shifting the k bits to the left by $(n - k)$ bits, and padding the result with zeroes (recall that left shift by 1 bit of a binary sequence is equivalent to multiplying the number represented by the binary sequence by two). The codeword, T , can then be represented as

$$T = 2^{n-k}D + F \quad (4.21)$$

Adding F in the above equation yields the concatenation of D and F . If we divide $2^{n-k}D$ by P , we obtain

$$\frac{2^{n-k}D}{P} = Q + \frac{R}{P} \quad (4.22)$$

where, Q is the quotient and R/P is the remainder. Suppose we use R as the FCS, then,

$$T = 2^{n-k}D + R \quad (4.23)$$

In this case, upon dividing T by P we obtain

$$\begin{aligned} \frac{T}{P} &= \frac{2^{n-k}D + R}{P} = \frac{2^{n-k}D}{P} + \frac{R}{P} \\ &= Q + \frac{R}{P} + \frac{R}{P} = Q + \frac{R+R}{P} = Q \end{aligned} \quad (4.24)$$

Thus there is no remainder, i.e., T is exactly divisible by P . To generate such an FCS we simply divide $2^{n-k}D$ by P and use the $(n - k)$ -bit remainder as the FCS.

Let an error E occur when T is transmitted over a noisy channel. The received word is given by

$$V = T + E \quad (4.25)$$

The CRC scheme will fail to detect the error only if V is completely divisible by P . This translates to the case when E is completely divisible by P (because T is divisible by P).

Example 4.24 Let the message $D = 1010001101$, i.e., $k = 10$ and the pattern, $P = 110101$. The

number of FCS bits = 5. Therefore, $n = 15$. We wish to determine the FCS.

First, the message is multiplied by 2^5 (left shift by 5 and pad with 5 zeroes). This yields

$$2^5 D = 101000110100000$$

Next divide the resulting number by $P = 110101$. By long division, we obtain $Q = 1101010110$ and $R = 01110$. The remainder is added to $2^5 D$ to obtain

$$T = 101000110101110$$

T is the transmitted codeword. If no errors occur in the channel, the received word, when divided by P , will yield 0 as the remainder.

CRC codes can also be defined using the polynomial representation. Let the message polynomial be $D(x)$ and the predetermined divisor be $P(x)$. Therefore,

$$\begin{aligned} \frac{x^{n-k} D(x)}{P(x)} &= Q(x) + \frac{R(x)}{P(x)} \\ T(x) &= x^{n-k} D(x) + R(x) \end{aligned} \quad (4.26)$$

At the receiver end, the received word is divided by $P(x)$. Suppose the received word is

$$V(x) = T(x) + E(x) \quad (4.27)$$

where $E(x)$ is the error polynomial. Then $[T(x) + E(x)]/P(x) = E(x)/P(x)$ because $T(x)/P(x) = 0$. Those errors that happen to correspond to polynomials containing $P(x)$ as a factor will slip by, and the others will be caught in the net of the CRC decoder. The polynomial $P(x)$ is also called the generator polynomial for the CRC code. CRC codes are also known as **polynomial codes**.

Example 4.25 Let $P(x) = x^{16} + x^{15} + x^2 + 1$. This $P(x)$ acts as the generator polynomial. Let the message to encode be $D(x) = x^{13} + x + 1$.

Since $n - k = \text{degree of } P(x) = 16$, first multiply $D(x)$ by x^{16} .

$$x^{16} D(x) = x^{19} + x^{17} + x^{16}$$

Divide $x^{16} D(x)$ by $P(x)$ to obtain

$$R(x) = x^{15} + x^5 + x^4 + x^3 + 1$$

The codeword will be

$$\begin{aligned} T(x) &= x^{16} D(x) + R(x) \\ &= (x^{19} + x^{17} + x^{16}) + (x^{15} + x^5 + x^4 + x^3 + 1) \\ &= x^{19} + x^{17} + x^{16} + x^{15} + x^5 + x^4 + x^3 + 1 \end{aligned}$$

Example 4.26 Suppose the transmitted codeword undergoes a single-bit error. The error polynomial $E(x)$ can be represented by $E(x) = x^i$, where i determines the location of the single-bit error. If $P(x)$ contains two or more terms, $E(x)/P(x)$ can never be zero. Thus all the single errors will be caught by such a CRC code.

Example 4.27 Suppose two isolated errors occur, i.e., $E(x) = x^i + x^j$, $i > j$. Alternately, $E(x) = x^j(x^{i-j} + 1)$. If we assume that $P(x)$ is not divisible by x , then a sufficient condition for detecting all double errors is that $P(x)$ does not divide $x^k + 1$ for any k up to the maximum value of $i - j$ (i.e., the frame length). For example $x^{15} + x^{14} + 1$ will not divide $x^k + 1$ for any value of k below 32,768.

Example 4.28 Suppose the error polynomial has an odd number of terms (corresponding to an odd number of errors). An interesting fact is that there is no polynomial with an odd number of terms that has $x + 1$ as a factor if we are performing binary arithmetic (modulo 2 operations). By making $(x + 1)$ as a factor of $P(x)$, we can catch all errors consisting of odd number of bits (i.e., we can detect at least half of all possible errors!).



Another interesting feature of CRC codes is its ability to detect burst errors. A burst error of length k can be represented by $x^i(x^{k-1} + x^{k-2} + \dots + 1)$, where i determines how far from the right end of the received frame the burst is located. If $P(x)$ has an x^0 term, it will not have an x^i as a factor. So, if the degree of $(x^{k-1} + x^{k-2} + \dots + 1)$ is less than the degree of $P(x)$, the remainder can never be zero. Therefore, a polynomial code with r check bits can detect all burst errors of length $\leq r$. If the burst length is $r + 1$, the remainder of the division by $P(x)$ will be zero if and only if the burst is identical to $P(x)$. Now, the 1st and last bits of a burst must be 1 (by definition). The intermediate bits can be 1 or 0. Therefore, the exact matching of the burst error with the polynomial $P(x)$ depends on the $r - 1$ intermediate bits. Assuming all combinations are equally likely, the probability of a miss is $\frac{1}{2^{r-1}}$. One can show that when error bursts of length greater than $r + 1$ occur, or several shorter bursts occur, the probability of a bad frame slipping through is $\frac{1}{2^r}$.

Example 4.29 Some of the popular of $P(x)$ have become international standards:

$$\text{CRC-4: } P(x) = x^4 + x^3 + x^2 + x^1 + 1$$

$$\text{CRC-7: } P(x) = x^7 + x^6 + x^4 + 1$$

$$\text{CRC-8: } P(x) = x^8 + x^7 + x^6 + x^4 + x^2 + 1$$

$$\text{CRC-12: } P(x) = x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$$

$$\text{CRC-16: } P(x) = x^{16} + x^{15} + x^2 + 1$$

$$\text{CRC-CCITT: } P(x) = x^{16} + x^{12} + x^5 + 1$$

$$\text{CRC-32: } P(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$$

$$\text{CRC-40 GSM: } P(x) = x^{40} + x^{26} + x^{23} + x^{17} + x^3 + 1$$

CRC-64 ISO: $P(x) = x^{64} + x^4 + x^3 + x^1 + 1$

CRC-64 ECMA-182: $P(x) = x^{64} + x^{62} + x^{57} + x^{55} + x^{54} + x^{53} + x^{52} + x^{47} + x^{46} + x^{45} + x^{40} + x^{39} + x^{38} + x^{37} + x^{35} + x^{33} + x^{32} + x^{31} + x^{29} + x^{27} + x^{24} + x^{23} + x^{22} + x^{21} + x^{19} + x^{17} + x^{13} + x^{12} + x^{10} + x^9 + x^7 + x^4 + x^1 + 1$ (4.28)

All the CRC codes listed above contain $(x + 1)$ as a factor. Suppose $P(x)$ has a factor $(x + 1)$. Then, it can be shown that all codewords have even parity. Thus it can detect any odd number of bits in error! CRC-12 is used for transmission of streams of 6-bit characters and generates a 12-bit FCS. Both CRC-16 and CRC-CCITT are popular for 8-bit characters. They result in a 16 bit FCS and can identify all single and double errors, all errors with odd number of bits, all burst errors of length 16 or less, 99.997% of 17-bit bursts and 99.998% of 18-bit and longer bursts.

INDUSTRIAL RELEVANCE



CRC-32 is specified as an option in some point-to-point **Synchronous Transmission Standards**. CRC-40 is used in the GSM control channel. CRC-64 ISO is used in the **High-Level Data Link Control (HDLC)** which is a bit-oriented code-transparent synchronous data link layer protocol developed by the International Organization for Standardization (ISO). CRC-64 ECMA-182 is a part of the **European Computer Manufacturers Association (ECMA)** standard.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 3:

- Consider a CRC code with $P(x) = x^{16} + x^{15} + x^2 + 1$. Factorise this polynomial and then determine how many odd number of errors it can detect? Also comment on the detection capability for double errors.
- Consider a CRC code with $P(x) = x^{16} + x^{15} + x^2 + 1$. Let the message polynomial be $D(x) = x^{14} + x^{13} + x^{11} + x^{10} + x^8 + x^5 + x^2 + x + 1$. Find the corresponding codeword polynomial, $T(x)$. Can this CRC code detect 7 errors?
- Consider a CRC-16 code with $P(x) = x^{16} + x^{15} + x^2 + 1$. What can be the largest size of the codeword (in bytes) that can detect double errors ?
- Consider the (24, 12) Golay code used in HF radio systems by NASA, for Automatic Link Establishment with $g(x) = x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1$. Give the corresponding generator matrix, G_{24} and the table of weights of different codewords.

S

M

M

M

If you have successfully solved the above problems,
you have mastered LO 3. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/598>



4.11 Circuit Implementation of Cyclic Codes

Shift registers can be used to encode and decode cyclic codes easily. Encoding and decoding of cyclic codes require multiplication and division by polynomials. The shift property of shift registers is ideally suited for such operations. Shift registers are banks of memory units which are capable of shifting the contents of one unit to the next at every clock pulse. Here we will focus on circuit implementation for codes over $GF(2^m)$. Beside the shift register, we will make use of the following circuit elements:

- (i) A scaler, whose job is to multiply the input by a fixed field element.
- (ii) An adder, which takes in two inputs and adds them together. A simple circuit realisation of an adder is the ‘exclusive-or’ or the ‘*xor*’ gate.
- (iii) A multiplier, which is basically the ‘*and*’ gate.

These elements are depicted in Fig. 4.2.

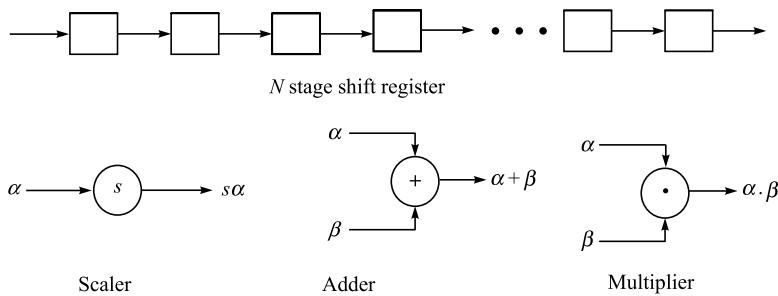


Fig. 4.2 Circuit elements used to construct encoders and decoders for cyclic codes.

A field element of $GF(2)$ can simply be represented by a single bit. For $GF(2^m)$ we require m bits to represent one element. For example, elements of $GF(8)$ can be represented as the elements of the set $\{000, 001, 010, 011, 100, 101, 110, 111\}$. For such a representation we need three clock pulses to shift the element from one *stage* of the shift register to the next. The effective shift register for $GF(8)$ is shown in Fig. 4.3. Any arbitrary element of this field can be represented by $ax^2 + bx + c$, where a, b, c are binary, and the power of the indeterminate x is used to denote the position. For example, $101 = x^2 + 1$.

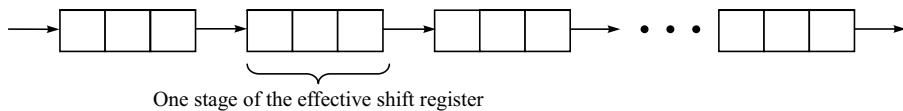


Fig. 4.3 The effective shift register for $GF(8)$.

LO 4



Explain how to implement cyclic codes using circuits.

Example 4.30 We now consider the multiplication of any arbitrary element by another field element over $GF(8)$. Recall the construction of $GF(8)$ from $GF(2)$ using the prime polynomial $p(x) = x^3 + x + 1$. The elements of the field will be $0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1$. We want to obtain the circuit representation for the multiplication of any arbitrary field element $(ax^2 + bx + c)$ by another element, say, $x^2 + x$. We have

$$\begin{aligned}(ax^2 + bx + c)(x^2 + x) &= ax^4 + (a+b)x^3 + (b+c)x^2 + cx \quad (\text{modulo } p(x)) \\ &= (a+b+c)x^2 + (b+c)x + (a+b)\end{aligned}$$

One possible circuit realisation is shown in Fig. 4.4.

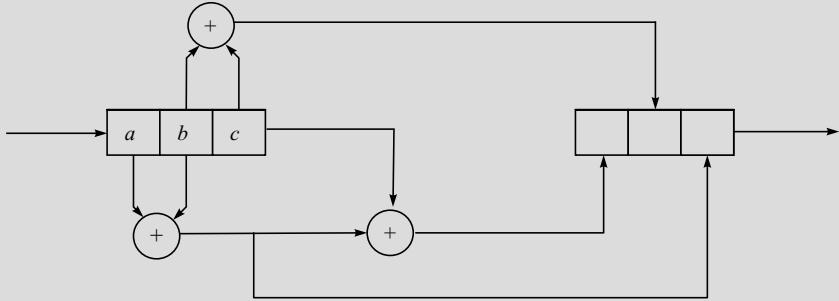


Fig. 4.4 Multiplication of an arbitrary field element $(ax^2 + bx + c)$ by the element $(x^2 + x)$ over $GF(8)$.

We next focus on the multiplication of an arbitrary polynomial $a(x)$ by $g(x)$. Let the polynomial $g(x)$ be represented as

$$g(x) = g_L x^L + \dots + g_1 x + g_0 \quad (4.29)$$

the polynomial $a(x)$ be represented as

$$a(x) = a_k x^k + \dots + a_1 x + a_0 \quad (4.30)$$

the resultant polynomial $b(x) = a(x)g(x)$ be represented as

$$b(x) = b_{k+L} x^{k+L} + \dots + b_1 x + b_0 \quad (4.31)$$

The circuit realisation of $b(x)$ is given in Fig. 4.5. This is linear feed-forward shift register. It is also called a **Finite Impulse Response (FIR)** filter.

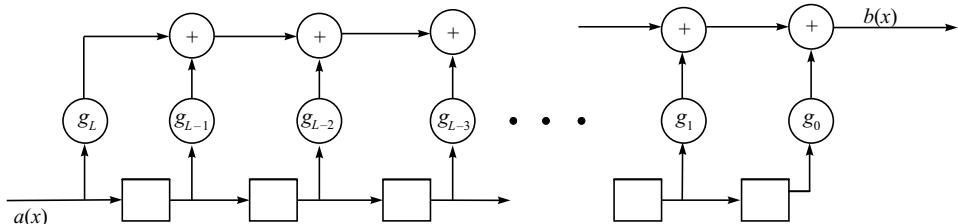


Fig. 4.5 A Finite Impulse Response (FIR) filter realisation of the product of two polynomials $b(x) = a(x)g(x)$.

In Electrical Engineering jargon, the coefficients of $a(x)$ and $g(x)$ are convolved by the shift register. For our purpose, we have a circuit realisation for multiplying two polynomials. Thus, we have an efficient mechanism of encoding a cyclic code by multiplying the information polynomial by the generator polynomial.

Example 4.31 The encoder circuit for the generator polynomial

$$g(x) = x^8 + x^6 + x^5 + x^3 + x + 1$$

is given in Fig. 4.6.

This is the generator polynomial for the Fire code with $t = m = 3$. It is easy to interpret the circuit. The 8 memory units shift the input, one unit at a time. The shifted outputs are summed at the proper locations. There are five adders for summing up the six shifted versions of the input.

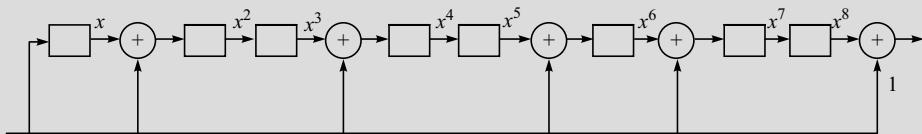


Fig. 4.6 Circuit realisation of the encoder for the Fire code with the generator polynomial
 $g(x) = x^8 + x^6 + x^5 + x^3 + x + 1$.

Example 4.32 Consider the systematic encoding process for the cyclic (7, 3) code using $g(x) = x^4 + x^3 + x^2 + 1$. The circuit is shown in Fig. 4.7. Observe the two switches in the circuit. The encoding is done in two steps.

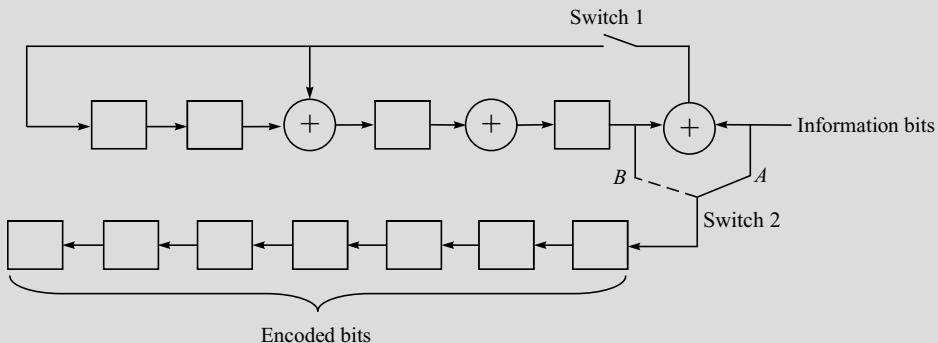


Fig. 4.7 A circuit for systematic encoding using $g(x) = x^4 + x^3 + x^2 + 1$.

Step 1: Switch 1 is in open position. Connect Switch 2 to 'A' in order to read in the information bits during the 1st three shifts. This being a systematic encoder, the 1st three codeword bits will be the information bits.

Step 2: Switch 1 is in closed position. Connect Switch 2 to 'B' in order to read in the encoded bits in the next four shifts.

We can also use a shift register circuit for dividing an arbitrary polynomial, $a(x)$, by a fixed polynomial $g(x)$. We assume here that the divisor is a monic polynomial. We already know how to factor out a scalar in order to convert any polynomial to a monic polynomial. The division process can be expressed as a pair of recursive equations. Let $Q^{(r)}(x)$ and $R^{(r)}(x)$ be the quotient polynomial and the remainder polynomial at the r^{th} recursion step, with the initial conditions $Q^{(0)}(x) = 0$ and $R^{(0)}(x) = a(x)$. Then, the recursive equations can be written as

$$\begin{aligned} Q^{(r)}(x) &= Q^{(r-1)}(x) + R_{(n-r)}x^{k-r} \\ R^{(r)}(x) &= R^{(r-1)}(x) - R_{(n-r)}x^{k-r}g(x) \end{aligned} \quad (4.32)$$

where $R_{(n-r)}$ represents the leading coefficient of the remainder polynomial at stage $(r-1)$. For dividing an arbitrary polynomial $a(x)$ by a fixed polynomial $g(x)$, the circuit realisation is given in Fig. 4.8. After n shifts, the quotient is passed out of the shift register, and the value stored in the shift register is the remainder. Thus the shift register implementation of a decoder is very simple. The contents of the shift register can be checked for all entries to be zero after the division of the received polynomial by the generator polynomial. If even a single memory unit of the shift register is non-zero, an error is detected.

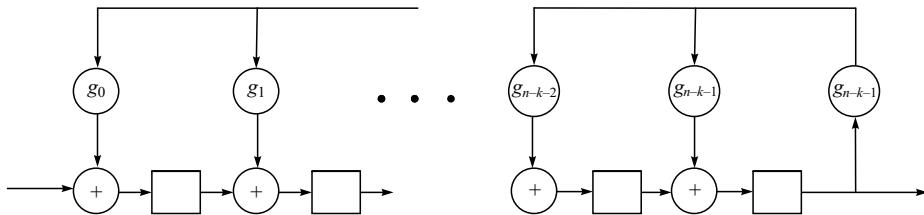


Fig. 4.8 A shift register circuit for dividing by $g(x)$.

Example 4.33

The circuit for dividing by $g(x) = x^3 + x + 1$ is given in Fig. 4.9.

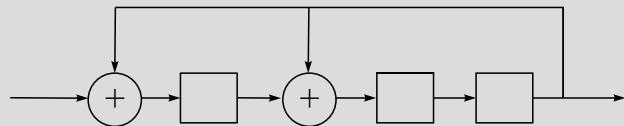


Fig. 4.9 Circuit realisation for dividing by $g(x) = x^3 + x + 1$.

Example 4.34 The circuit for dividing by $g(x) = x^4 + x^2 + 1$ is given in Fig. 4.10.

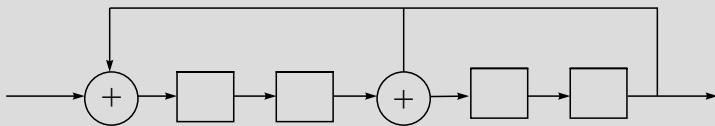


Fig. 4.10 Circuit realisation for dividing by $g(x) = x^4 + x^2 + 1$.

Example 4.35 The shift register circuit for dividing by $g(x) = x^8 + x^6 + x^5 + x^3 + x + 1$ is given in Fig. 4.11.

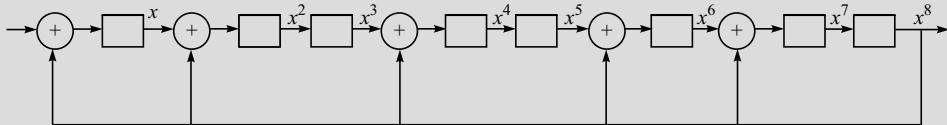


Fig. 4.11 A shift register circuit for dividing by $g(x) = x^8 + x^6 + x^5 + x^3 + x + 1$.

The procedure for error detection and error correction is as follows. The received word is first stored in a buffer. It is subjected to divide-by- $g(x)$ operation. As we have seen, the division can be carried out very efficiently by a shift register circuit. The remainder in the shift register is then compared with all the possible (pre-computed) syndromes. This set of syndromes corresponds to the set of correctable error patterns. If a syndrome match is found, the error is subtracted out from the received word. The corrected version of the received word is then passed on to the next stage of the receiver unit for further processing. This kind of a decoder is known as the **Meggitt Decoder**. The flow chart for a decoder is given in Fig. 4.12.

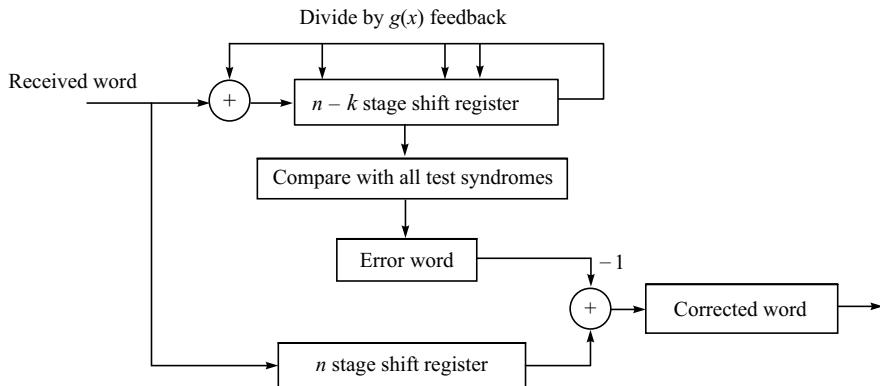


Fig. 4.12 The flowchart of a Meggitt Decoder.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 4:

- Design a circuit for error detection for the binary cyclic (7, 3) code using

$$g(x) = x^4 + x^3 + x^2 + 1$$

- Determine the generator polynomial for the circuit shown in Fig. 4.13. The encoding is done in two steps. In step 1, Switch 1 is in open position and we connect Switch 2 to 'A' in order to read in the information bits during the 1st four shifts. In step 2, Switch 1 is in closed position and we connect Switch 2 to 'B' in order to read in the encoded bits in the next three shifts.

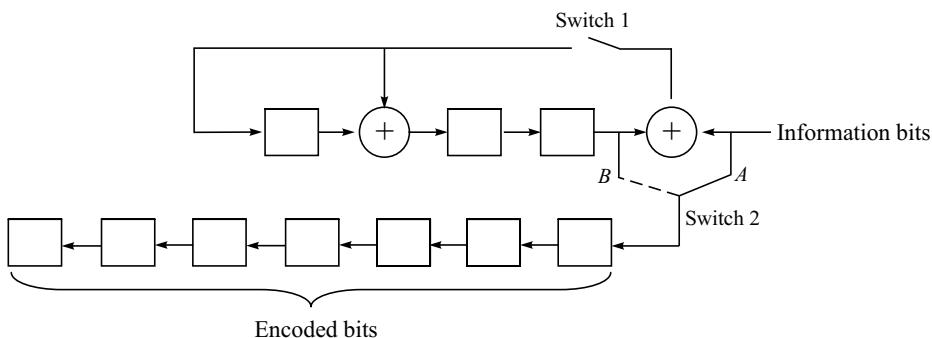


Fig. 4.13

If you have successfully solved the above problems,
you have mastered LO 4. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/599>



4.12 Concluding Remarks

Cyclic codes are interesting because of two primary reasons: (i) the structure of polynomial rings leads to structure formation in the code and (ii) the existence of elegant mathematical tools based on polynomials that can be used to analyse these codes. The notion of cyclic codes was first introduced by Prange in 1957. The work on cyclic codes was further developed by Peterson and Kasami. Pioneering work on the minimum distance of cyclic codes was done by Bose and Raychaudhuri in the early 1960s. Another subclass of cyclic codes, the BCH codes (named after Bose, Chaudhuri and Hocquenghem) will be studied in detail in the next chapter. It was soon discovered that almost all the LBCs discovered earlier could be converted to cyclic. The initial steps in the area of burst error correction were taken by Abramson in 1959. The Fire codes were published in the same year. The binary and the ternary Golay codes were published by Golay in, as early as, 1949.

Shift register circuits for cyclic codes were introduced in the works of Peterson, Chien and Meggitt in the early 1960s. Important contributions were also made by Kasami, MacWilliams, Mitchell and Rudolph.

LEARNING OUTCOMES

- A polynomial is a mathematical expression $f(x) = f_0 + f_1x + \dots + f_mx^m$, where the symbol x is called the indeterminate and the coefficients f_0, f_1, \dots, f_m are the elements of $GF(q)$. The coefficient f_m is called the leading coefficient. If $f_m \neq 0$, then m is called the degree of the polynomial, and is denoted by $\deg f(x)$. A polynomial is called monic if its leading coefficient is unity.
- The division algorithm states that, for every pair of polynomials $a(x)$ and $b(x) \neq 0$ in $F[x]$, there exists a unique pair of polynomials $q(x)$, the quotient, and $r(x)$, the remainder, such that $a(x) = q(x)b(x) + r(x)$, where $\deg r(x) < \deg b(x)$. The remainder is sometimes also called the residue, and is denoted by $R_{b(x)}[a(x)] = r(x)$.
- Two important properties of residues are as follows:
 - (i) $R_{f(x)}[a(x) + b(x)] = R_{f(x)}[a(x)] + R_{f(x)}[b(x)]$, and
 - (ii) $R_{f(x)}[a(x) \cdot b(x)] = R_{f(x)}\{R_{f(x)}[a(x)] \cdot R_{f(x)}[b(x)]\}$
 where $a(x)$, $b(x)$ and $f(x)$ are polynomials over $GF(q)$.
- A polynomial $f(x)$ in $F[x]$ is said to be reducible if $f(x) = a(x)b(x)$, where $a(x)$, $b(x)$ are elements of $f(x)$ and $\deg a(x)$ and $\deg b(x)$ are both smaller than $\deg f(x)$. If $f(x)$ is not reducible, it is called irreducible. A monic irreducible polynomial of degree at least one is called a prime polynomial.
- The ring $F[x]/f(x)$ is a field if and only if $f(x)$ is a prime polynomial in $F[x]$.
- A code \mathbf{C} in R_n is a cyclic code if and only if \mathbf{C} satisfies the following conditions:
 - (i) $a(x), b(x) \in \mathbf{C} \Rightarrow a(x) + b(x) \in \mathbf{C}$,
 - (ii) $a(x) \in \mathbf{C}$ and $r(x) \in R_n \Rightarrow a(x)r(x) \in \mathbf{C}$.
- Following steps can be used to generate a cyclic code:
 - (i) Take a polynomial $f(x)$ in R_n .
 - (ii) Obtain a set of polynomials by multiplying $f(x)$ by all possible polynomials in R_n .
 - (iii) The set of polynomials obtained as above corresponds to the set of codewords belonging to a cyclic code. The block length of the code is n .
- Let \mathbf{C} be a (n, k) non-zero cyclic code in R_n . Then,
 - (i) there exists a unique monic polynomial $g(x)$ of the smallest degree in C ,
 - (ii) the cyclic code C consists of all multiples of the generator polynomial $g(x)$ by polynomials of degree $k - 1$ or less.
 - (iii) $g(x)$ is a factor of $x^n - 1$.
 - (iv) The degree of $g(x)$ is $n - k$.
- For a cyclic code, \mathbf{C} , with generator polynomial $g(x) = g_0 + g_1x + \dots + g_rx^r$ of degree r , the generator matrix of given by

$$\boxed{\mathbf{G} = \begin{bmatrix} g_0 & g_1 & \dots & g_r & 0 & 0 & 0 & \dots & 0 \\ 0 & g_0 & g_1 & \dots & g_r & 0 & 0 & \dots & 0 \\ 0 & 0 & g_0 & g_1 & \dots & g_r & 0 & \dots & 0 \\ \vdots & \vdots & & & & & & \ddots & \\ 0 & 0 & 0 & 0 & 0 & g_0 & g_1 & \dots & g_r \end{bmatrix} \quad \begin{array}{c} k = (n - r) \text{ rows} \\ \uparrow \downarrow \\ n \text{ columns} \end{array}}$$

- For a cyclic code, \mathbf{C} , with the parity check polynomial $h(x) = h_0 + h_1x + \dots + h_kx^k$, then the parity check matrix is given by

$$\boxed{\mathbf{H} = \begin{bmatrix} h_k & h_{k-1} & \dots & h_0 & 0 & 0 & 0 & \dots & 0 \\ 0 & h_k & h_{k-1} & \dots & h_0 & 0 & 0 & \dots & 0 \\ 0 & 0 & h_k & h_{k-1} & \dots & h_0 & 0 & \dots & 0 \\ \vdots & \vdots & & & & & & \ddots & \\ 0 & 0 & 0 & 0 & 0 & h_k & h_{k-1} & \dots & h_0 \end{bmatrix} \quad \begin{array}{c} (n - k) \text{ rows} \\ \uparrow \downarrow \\ n \text{ columns} \end{array}}$$

- $x^n - 1 = h(x)g(x)$, where, $g(x)$ is the generator polynomial and $h(x)$ is the parity check polynomial.
- An (n, k) quasi-cyclic code is a linear block code such that for some m , co-prime with n , the polynomial $x^m c(x)$ calculated mod $x^n - 1$ is a codeword polynomial if $c(x)$ is a valid codeword polynomial.
- An (n, k) linear code is called a proper shortened cyclic code if it is obtained by deleting m consecutive places from an $(n + m, k + m)$ cyclic code.
- A Fire code is a cyclic burst error correcting code over $GF(q)$ with the generator polynomial $g(x) = (x^{2t-1} - 1)p(x)$, where $p(x)$ is a prime polynomial over $GF(q)$ whose degree m is not smaller than t and $p(x)$ does not divide $x^{2t-1} - 1$. The block length of the Fire code is the smallest integer n such that $g(x)$ divides $x^n - 1$. A Fire code can correct all burst errors of length t or less.
- The generator polynomial of the binary Golay Code:

$$\begin{aligned} g_1(x) &= (x^{11} + x^{10} + x^6 + x^5 + x^4 + x^2 + 1), \text{ or} \\ g_2(x) &= (x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1). \end{aligned}$$

- The generator polynomial of the ternary Golay Code:

$$\begin{aligned} g_1(x) &= (x^5 + x^4 - x^3 + x^2 - 1), \text{ or} \\ g_2(x) &= (x^5 - x^3 - x^2 - x - 1). \end{aligned}$$
- One of the common error detecting codes is the CRC codes. For a k -bit block of bits the (n, k) CRC encoder generates $(n - k)$ bit long frame check sequence (FCS).
- Shift registers can be used to encode and decode cyclic codes easily. Encoding and decoding of cyclic codes require multiplication and division by polynomials. The shift property of shift registers is ideally suited for such operations.

Everything should be made as simple as possible, but not simpler

Albert Einstein (1879–1955)



MULTIPLE CHOICE QUESTIONS

- 4.1 For the generation of a cyclic code, the generator polynomial should be the factor of
 - (a) $x^{n+1} + 1$
 - (b) $x^n - 1$
 - (c) $x^n / 2$
 - (d) $x^{2n/3}$
- 4.2 For residues, which of the following is true?
 - (a) $R_{f(x)}[a(x) + b(x)] = R_{f(x)}[a(x)] + R_{f(x)}[b(x)]$, and
 - (b) $R_{f(x)}[a(x) \cdot b(x)] = R_{f(x)}\{R_{f(x)}[a(x)] \cdot R_{f(x)}[b(x)]\}$
 - (c) Both (a) and (b)
 - (d) None of the above
- 4.3 Which of the following is true?
 - (a) A ring $F[x]/f(x)$ is a field if and only if $f(x)$ is a monic polynomial in $F[x]$
 - (b) A ring $F[x]/f(x)$ is a field if and only if $f(x)$ is a prime polynomial in $F[x]$
 - (c) A ring $F[x]/(f(x) + 1)$ is a field if and only if $f(x)$ is a prime polynomial in $F[x]$
 - (d) None of the above
- 4.4 A code \mathbf{C} in R_n is a cyclic code if and only if \mathbf{C} satisfies
 - (a) $a(x), b(x) \in \mathbf{C} \Rightarrow a(x) + b(x) \in \mathbf{C}$
 - (b) $a(x) \in \mathbf{C}$ and $r(x) \in R_n \Rightarrow a(x)r(x) \in \mathbf{C}$
 - (c) Both (a) and (b)
 - (d) None of the above
- 4.5 The degree of $g(x)$ is
 - (a) $n - k + 1$
 - (b) $n + k + 1$
 - (c) $n - k - 1$
 - (d) $n - k$
- 4.6 If $x^n - 1 = h(x)g(x)$, where, $g(x)$ is the generator polynomial, then the parity check polynomial would be
 - (a) $h(x)$
 - (b) $g(x)h(x)$
 - (c) x^n
 - (d) $x - 1$
- 4.7 An (n, k) linear code is called a proper shortened cyclic code if it is obtained by deleting m consecutive places from an
 - (a) $(n - m, k + m)$ cyclic code
 - (b) $(n + m, k - m)$ cyclic code
 - (c) $(n + m, k + m)$ cyclic code
 - (d) None of the above
- 4.8 Which of the following is capable of correcting any combination of three or fewer random errors in a block of 23 bits?
 - (a) Hamming code
 - (b) Shortened Hamming code
 - (c) CRC code
 - (d) Golay code

4.9 Which is the generator polynomial of the binary Golay code:

- (a) $g_1(x) = (x^{11} + x^{10} + x^6 + x^5 + x^4 + x^2 + 1)$, or
- (b) $g_2(x) = (x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1)$
- (c) Both (a) and (b)
- (d) None of the above

4.10 Which is the generator polynomial of the ternary Golay code:

- (a) $g_1(x) = (x^5 + x^4 - x^3 + x^2 - 1)$, or
- (b) $g_2(x) = (x^5 - x^3 - x^2 - x - 1)$
- (c) Both (a) and (b)
- (d) None of the above

For interactive quiz with answers, scan the QR code given here

Or

Visit <http://qrcode.flipick.com/index.php/569>



SHORT-ANSWER TYPE QUESTIONS



- 4.1 No element of ring can have a multiplicative inverse. True or false?
- 4.2 Is the $(n, 1)$ repetition code a cyclic code? Is the $(7, 4)$ binary Hamming Code cyclic?
- 4.3 The generator matrix of a cyclic code can be represented as a Toeplitz matrix. True or false?
- 4.4 The parity check polynomial of a cyclic code can be used to generate another cyclic code. True or false?
- 4.5 Give a generator polynomial (of degree 4) for the $(15, 11)$ binary Hamming Code.
- 4.6 Suppose the generator polynomial for a cyclic code is $g(x) = g_0 + g_1x + \dots + g_rx^r$. Can g_0 be 0?
- 4.7 If $g(x)$ has a factor $(x + 1)$ then no codeword has odd weight. True or false?
- 4.8 For the $(15, 11)$ cyclic code, if $g(x) = x^4 + x + 1$, find parity check polynomial $h(x)$.
- 4.9 For a cyclic code, if $g(x) = x^{n-1} + x^{n-2} + \dots + x + 1$, find parity check polynomial $h(x)$.
- 4.10 If $g(x)$ is a generator polynomial of an (n, k) , then $g(x^{\gamma})$ is a generator polynomial of an $(\gamma n, \gamma k)$ code. True or false?
- 4.11 The minimum weights of the cyclic code generated by $g(x)$ and by the cyclic code generated by $g(x^g)$ are the same. True or false?
- 4.12 What is the code rate of the cyclic code of $n = 7$ with $g(x) = x + 1$?
- 4.13 How many errors can the cyclic code given by $g(x) = x^8 + x^7 + x^6 + x^4 + 1$ correct?
- 4.14 Is the code generated by $\mathbf{G} = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}$ cyclic?
- 4.15 Find the dual code for the $(9, 6)$ cyclic code given by $g(x) = x^3 + 1$.

For answers, scan the QR code given here

Or

Visit <http://qrcode.flipick.com/index.php/600>





PROBLEMS

- S** 4.1 Which of the following codes are (a) cyclic, (b) equivalent to a cyclic code?
- {0000, 0110, 1100, 0011, 1001} over $GF(2)$.
 - {00000, 10110, 01101, 11011} over $GF(2)$.
 - {00000, 10110, 01101, 11011} over $GF(3)$.
 - {0000, 1122, 2211} over $GF(3)$.
 - The q -ary repetition code of length n .
- M** 4.2 Construct the addition and multiplication table for
- $F[x]/(x^2 + 1)$ defined over $GF(2)$.
 - $F[x]/(x^2 + 1)$ defined over $GF(3)$.
- Which of the above is a field?
- S** 4.3 Answer the following for polynomials over $GF(16)$:
- How many distinct second-degree monic polynomials of the form $x^2 + ax + b$ ($b \neq 0$) exist?
 - How many distinct polynomials of the form $(x - \alpha)(x - \beta)$ exist?
- S** 4.4 List out all the irreducible polynomials over
- $GF(2)$ of degrees 1 to 5.
 - $GF(3)$ of degrees 1 to 3.
- S** 4.5 Find all the cyclic binary codes of block length 5. Find is the minimum distance of each code.
- S** 4.6 Suppose $x^n - 1$ is a product of r distinct irreducible polynomials over $GF(q)$. How many cyclic codes of block length n over $GF(q)$ exist? What can you comment on the minimum distance of these codes?
- S** 4.7 (i) Factorise $x^8 - 1$ over $GF(3)$.
- How many ternary cyclic codes of length 8 exist?
 - How many quaternary cyclic codes of length 8 exist?
- M** 4.8 Let the polynomial
- $$g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$$
- be the generator polynomial of a cyclic code over $GF(2)$ with block length 15.
- Find the generator polynomial \mathbf{G} .
 - Find the parity check matrix \mathbf{H} .
 - How many errors can this code detect?
 - How many errors can this code correct?
 - Write the generator matrix in the systematic form.
- D** 4.9 Consider the polynomial
- $$g(x) = x^6 + 3x^5 + x^4 + x^3 + 2x^2 + 2x + 1$$
- Is this a valid generator polynomial for a cyclic code over $GF(4)$ with block length 15?
 - Find the parity check matrix \mathbf{H} .
 - What is the minimum distance of this code?
 - What is the code rate of this code?
 - Is the received word $v(x) = x^8 + x^5 + 3x^4 + x^3 + 3x + 1$ a valid codeword?

- S** 4.10 Is $g(x) = x^{10} + x^7 + x^6 + x^4 + x^2 + 1$ a valid generator polynomial for a binary cyclic code with $n = 21$? Why/why not?
- M** 4.11 Consider a binary cyclic code with $n = 15$ and a generator polynomial given by $g(x) = x^8 + x^7 + x^6 + x^4 + 1$. Find $h(x)$ and the parity check matrix.
- D** 4.12 An error vector of the form $x^i + x^{i+1}$ in R_n is called a double adjacent error. Show that the code generated by the generator polynomial $g_1(x) = (x - 1)g_H(x)$ is capable of correcting all double adjacent errors, where $g_H(x)$ is the generator polynomial of the binary Hamming code.
- S** 4.13 Determine the burst error correction length for the cyclic code represented by the following generator polynomial
- $$g(x) = x^{13} + x^{10} + x^7 + x^6 + x^5 + x^4 + x^2 + 1$$
- S** 4.14 Show that the block length of a Fire code is $n = \text{LCM}(2t - 1, q^m - 1)$.
- S** 4.15 Show that $g(x) = (x^{11} + 1)(x^6 + x + 1)$ is a valid generator polynomial of a Fire code. What is the burst error correction length for this code?
- S** 4.16 Show that the cyclic code given by $g(x) = x^{14} + x^{11} + x^9 + x^5 + x^2 + 1$ is capable of correcting a burst error of length 5 or less.
- S** 4.17 Show that a shortened binary cyclic code with generator polynomial of degree $n - k$ can detect a fraction $1 - 2^{-(n-k)+1}$ of all burst patterns of length $(n - k) + 1$.
- M** 4.18 Design the shift register encoder and the Meggitt decoder for the code generated in Problem 4.9.
- M** 4.19 The code with the generator polynomial $g(x) = (x^{23} + 1)(x^{17} + x^3 + 1)$ is used for error-detection and correction in the GSM standard.
 - How many random errors can this code correct?
 - How many burst errors can this code correct?
- M** 4.20 Consider the (23, 12) binary Golay code given by $g(x) = x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1$. The first row of the generator matrix for this code will be $[1 \ 0 \ 1 \ 0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$. The other 11 rows are simply the right shifts of this 23-tuple. Find the weight distribution of the code.
- M** 4.21 Consider a binary cyclic code with the generator polynomial $g(x)$ and block length n .
 - If $(x + 1)$ is a factor of $g(x)$, show that all codewords must be of even weight.
 - If $(x + 1)$ is *not* a factor of $g(x)$ and n is odd, show that the code must contain the all-one codeword 111....1.

COMPUTER PROBLEMS



- Write a computer program to find the minimum distance of a cyclic code over $GF(q)$, given the generator polynomial (or the generator matrix) for the code.
- Write a computer program to divide a polynomial $a(x)$ by another polynomial $b(x)$ over $GF(q)$. The program should return the quotient and the residue.
- Write a computer program to factorise $x^n - 1$ into prime polynomials over $GF(2)$. Can you generalise this over $GF(q)$? The program should return the coefficients and the degrees of the prime polynomials.

- 4.4 Consider $g(x) = (x + 1)(x^a + x^b + 1)$. Let the codeword length, n , be a design parameter. Write a computer program to find a set of pairs $\{a, b\}$ such that $g(x)$ has good double error detection capability for large n .
- 4.5 Write a computer program to encode and decode a (35, 27) Fire code. It should be able to automatically correct bursts of length 3 or less. What happens when you try to decode a received word with a burst error of length 4? Write another computer program to encode and decode a (19437, 19408) Fire code. What is the burst error correction length for this code?
- 4.6 Write a computer program to search for cyclic Hamming codes over $GF(5)$. Is there a cyclic Hamming code of block length $n = 156$ over this Galois Field?
- 4.7 The IEEE 802.16 broadband wireless access group prescribes the following polynomial for CRC

$$g(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1.$$
(i) What error patterns can this code detect?
(ii) What is the burst error correction length for this code?

PROJECT IDEAS

- 4.1 **A Communication System with Golay Code:** Simulate a wireless communication system using a Golay encoder, a Fading Channel and a Golay decoder. The aim of the project is to evaluate the performance of a binary (23, 12) Golay Code and a ternary (11, 6) Golay Code over a slow-fading Rayleigh channel. Specifically, provide the following in both cases for a range of SNRs:
(i) probability of word error,
(ii) probability of bit error,
(iii) probability of undetected codeword error.
- 4.2 **Reciprocal:** Consider a cyclic code with the generator polynomial $g(x)$. Define the reciprocal of $g(x)$ as $g^*(x) = x^{n-k} g(1/x)$.
(i) Find the reciprocal of the Hamming code given by $g(x) = x^3 + x + 1$.
(ii) Show that $g^*(x)$ is also cyclic.
(iii) Show that the weight distribution of the codes generated by $g(x)$ and $g^*(x)$ are the same.
(iv) Repeat the above steps for

$$g_1(x) = (x^{11} + x^{10} + x^6 + x^5 + x^4 + x^2 + 1) \text{ and}$$

$$g_2(x) = (x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1)$$
- 4.3 **Meggitt Decoder:** Design a Meggitt Decoder that can handle both binary and ternary cyclic codes. Comment on the computational complexity of this decoder. Assuming realistic values for each circuit operation (addition, multiplication, comparison, shift register operation etc.) in μJ per operation, give the average energy estimate for decoding (in $\mu J/\text{bit}$).
- 4.4 **Weight Distribution of Cyclic Codes:** Make a table of the weights of different codewords and how many codewords are there for that weight. Carry out this exercise for the following:
(i) Binary (23, 12) Golay Code
(ii) Ternary (11, 6) Golay Code
(iii) Binary (35, 27) Fire code
(iv) Binary (19437, 19408) Fire code

REFERENCES FOR FURTHER READING

1. Roth, R.M., *Introduction to Coding Theory*, Cambridge University Press, 2006.
2. Lin, S. and Costello, D.J., *Error Control Coding: Fundamentals and Applications* (2nd edition), Prentice-Hall, 2004.
3. Blahut, R.E., *Algebraic Codes for Data Transmission*, Cambridge University Press, 2002.
4. Moon, T.K., *Error Correction Coding: Mathematical Methods and Algorithms*, Wiley, 2005.

Bose–Chaudhuri Hocquenghem (BCH) Codes

Although this may seem a paradox, all exact science is dominated by the idea of approximation.

Bertrand Russell (1872-1970)

Learning objectives

After studying this chapter the students will be able to:

- LO 1** Use the mathematical tools developed, including the ideas of a primitive element and minimal polynomials, to study Bose-Chaudhuri Hocquenghem (BCH) codes.
- LO 2** Define BCH codes, learn how to obtain generator polynomials for BCH codes and understand how to efficiently carry out BCH decoding.
- LO 3** Explain the powerful Reed-Solomon (RS) codes and understand how RS encoders and decoders work.

5.1 Introduction to BCH Codes

The class of **Bose-Chaudhuri Hocquenghem** (BCH) codes is one of the most powerful classes of linear cyclic block codes. BCH codes are known for their multiple error correcting ability, and the ease of encoding and decoding. So far, our approach has been to construct a code and then find out its minimum distance in order to estimate its error correcting capability. In this class of codes, we will start from the other end. We begin by specifying the number of random errors desired to be corrected by the code. Then we go on to construct the generator polynomial for the code. As mentioned above, BCH codes are a subclass of cyclic codes, and therefore, the decoding methodology for any cyclic code also works for the BCH codes. However, more efficient decoding procedures are known for BCH codes, and will be discussed in this chapter. This chapter will also introduce the **Reed-Solomon (RS) codes**.

...
This chapter comes with a video overview by the author. Scan here to know more or
Visit <http://qrcode.flipick.com/index.php/606>



In this chapter

We will first develop the necessary mathematical tools to study BCH codes, including the important idea of a primitive element of a Galois field and minimal polynomials, in LO 1.

Define BCH codes and learn how to obtain generator polynomials in terms of minimal polynomials. We will also look at efficient BCH decoding, in LO 2.

Finally, an important sub-set of BCH codes, the RS codes, will be introduced in the later part of this chapter. We will study these powerful RS codes and understand how RS encoders and decoders work, in LO 3.

5.2 Primitive Elements

LO 1



Use the mathematical tools developed, including the ideas of a primitive element and minimal polynomials, to study Bose-Chaudhuri Hocquenghem (BCH) codes.

We have already studied the basic properties of fields in Chapter 3. In layman's language, the elements of a field can be added, multiplied and divided. We also know that $GF(q)$ exists for an integer q which is either a prime or a prime power. All elements of a field have an additive inverse and a multiplicative inverse (except '0').

Example 5.1

$GF(5)$ and $GF(8)$ exist but $GF(6)$ and $GF(10)$ do not exist.

Definition 5.1 A **subfield** of a field is a subset of the field, which is also a field. Thus, the subfield is 'contained' in the original field. Alternately, we can say that there is a **base field**, which is contained in the **extension field**. The base field is also called a **ground field**.

As we shall soon discover, it is fairly easy to construct an extension field from a base field.

Definition 5.2 A **Primitive Element** of $GF(q)$ is an element α such that every field element, except zero, can be expressed as a power of α .

Example 5.2 Consider $GF(5)$. Since $q = 5$ = a prime number, modulo arithmetic will work.

Consider the element 2.

$$\begin{aligned} 2^0 &= 1 \pmod{5} = 1, \\ 2^1 &= 2 \pmod{5} = 2, \\ 2^2 &= 4 \pmod{5} = 4, \\ 2^3 &= 8 \pmod{5} = 3. \end{aligned}$$

Hence, all the elements of $GF(5)$ i.e., $\{1, 2, 3, 4, 5\}$ can be represented as powers of 2. Therefore, 2 is a primitive element of $GF(5)$.

Next, consider the element 3.

$$\begin{aligned}3^0 &= 1 \pmod{5} = 1, \\3^1 &= 3 \pmod{5} = 3, \\3^2 &= 9 \pmod{5} = 4, \\3^3 &= 27 \pmod{5} = 2.\end{aligned}$$

Again, all the elements of $GF(5)$ i.e., $\{1, 2, 3, 4, 5\}$ can be represented as powers of 3. Therefore, 3 is also a primitive element of $GF(5)$.

However, it can be tested that the other non-zero elements $\{1, 4, 5\}$ are not primitive elements.

Example 5.3 Consider $GF(4)$ with elements $\{0, 1, x, x + 1\}$.

Table 5.1 Addition and multiplication tables for $GF(4)$ with elements $\{0, 1, x, x + 1\}$

+	0	1	x	$x + 1$
0	0	1	x	$x + 1$
1	1	0	$x + 1$	x
x	x	$x + 1$	0	1
$x + 1$	$x + 1$	x	1	0

•	0	1	x	$x + 1$
0	0	0	0	0
1	0	1	X	$x + 1$
x	0	X	$x + 1$	1
$x + 1$	0	$x + 1$	1	x

Let $\alpha = x$.

$$\alpha^2 = x \cdot x = x + 1. \text{ This is from the multiplication table.}$$

$$\alpha^3 = x \cdot (x + 1) = 1. \text{ Again, this is from the multiplication table.}$$

Since all elements of $GF(4)$, except '0' can be expressed as a power of x , it is a primitive element.

Next, try $\alpha = x + 1$.

$$\alpha^2 = (x + 1) \cdot (x + 1) = x. \text{ This is from the multiplication table.}$$

$$\alpha^3 = (x + 1) \cdot x = 1. \text{ Again, this is from the multiplication table.}$$

Since all elements of $GF(4)$, except '0' can be expressed as a power of $(x + 1)$, it is also a primitive element.

However, one can see that one is not a primitive element (no pun intended)!

Also note that $GF(2)$ is a subfield of $GF(4)$. The addition and multiplication tables of $GF(2)$ are in fact embedded in the tables of $GF(4)$. This is clear if we just focus on the elements $\{0, 1\}$. Thus, we can say that $GF(2)$ is the base field and $GF(4)$ is the extension field.



We saw in the previous example that there can be more than one primitive element in a field. But is there a guarantee of finding at least one primitive element? The answer is yes! The non-zero elements of every Galois Field form a cyclic group. Hence, a Galois Field will include an element of order $q - 1$. This will be the primitive element. Primitive elements are very useful in constructing fields. Once we have a primitive element, we can easily find all the other elements by simply evaluating the powers of the primitive element. Since primitive elements exist in all fields, one can write all the non-zero elements of a field in terms of the powers of the primitive element. If α is a primitive element of $GF(q)$, then all the elements in the field are simply $\{0, \alpha, \alpha^2, \alpha^3, \dots, \alpha^{q-1}\}$.

Definition 5.3 For an element $\beta \in GF(q)$, let n be the smallest integer such that $\beta^n = 1$, then n is the **order** of β . The order of a primitive element is $q - 1$.

Example 5.4 Consider $GF(5)$.

Since, $2^4 = 1$, the order of 2 is 4.

Since, $3^4 = 1$, the order of 3 is 4.

Since, $4^2 = 1$, the order of 4 is 2.

Here, $q = 5$. Since both 2 and 3 are primitive elements, their order is $q - 1 = 4$.

Definition 5.4 A **Primitive Polynomial** $p(x)$ over $GF(q)$ is a prime polynomial over $GF(q)$ with the property that in the extension field constructed modulo $p(x)$, the field element represented by x is a primitive element.

Primitive polynomials of every degree exist over every Galois Field. A primitive polynomial can be used to construct an extension field.

Example 5.5 We can construct $GF(8)$ using the primitive polynomial $p(x) = x^3 + x + 1$. Let the primitive element of $GF(8)$ be α . Then, we can represent all the elements of $GF(8)$ by the powers of α evaluated modulo $p(x)$. Thus, we can form the Table 5.2.

Table 5.2 The elements of $GF(8)$

Powers of α	Elements of $GF(8)$
α^1	z
α^2	z^2
α^3	$z + 1$
α^4	$z^2 + z$
α^5	$z^2 + z + 1$
α^6	$z^2 + 1$
α^7	1

Theorem 5.1 Let $\beta_1, \beta_2, \dots, \beta_{q-1}$ denote the non-zero field elements of $GF(q)$. Then,

$$x^{q-1} - 1 = (x - \beta_1)(x - \beta_2) \dots (x - \beta_{q-1}) \quad (5.1)$$

Proof: The set of non-zero elements of $GF(q)$ is a finite group under the operation of multiplication. Let β be any non-zero element of the field. It can be represented as a power of the primitive element α . Let $\beta = (\alpha)^r$ for some integer r .

Therefore, $\beta^{q-1} = ((\alpha)^r)^{q-1} = ((\alpha)^{q-1})^r = (1)^r = 1$ because, $(\alpha)^{q-1} = 1$.

Hence, β is a zero of $x^{q-1} - 1$. This is true for *any* non-zero element β .

Hence, $x^{q-1} - 1 = (x - \beta_1)(x - \beta_2) \dots (x - \beta_{q-1})$.

Example 5.6 Consider the field $GF(5)$. The non-zero elements of this field are $\{1, 2, 3, 4\}$.

Therefore, we can write

$$x^4 - 1 = (x - 1)(x - 2)(x - 3)(x - 4)$$

Example 5.7 Consider the field $GF(8)$. From an earlier example, the non-zero elements of this field are $\{1, z, z^2, z + 1, z^2 + z, z^2 + z + 1, z^2 + 1\}$. Therefore, we can write

$$x^7 - 1 = (x - 1)(x - z)(x - z^2)(x - z - 1)(x - z^2 - z)(x - z^2 - z - 1)(x - z^2 - 1)$$

If we patiently carry out the multiplications and simplifications on the right-hand-side, all the z 's will magically cancel out, leaving a neat $(x^7 - 1)$ at the end! It is interesting to observe that we have before us an elegant way of factorising polynomials of the type $(x^{q-1} - 1)$, where q is a prime or a prime power. The question is: how do we use it to our advantage?

5.3 Minimal Polynomials

LO 1

In the previous chapter, we saw that in order to find the generator polynomials for cyclic codes of blocklength n , we have to first factorise $x^n - 1$. Thus, $x^n - 1$ can be written as the product of its p prime factors

$$x^n - 1 = f_1(x)f_2(x)f_3(x)\dots f_p(x) \quad (5.2)$$

Any combination of these factors can be multiplied together to form a generator polynomial $g(x)$. If the prime factors of $x^n - 1$ are distinct, then there are $(2^p - 2)$ different non-trivial cyclic codes of blocklength n . The two trivial cases that are being disregarded are $g(x) = 1$ and $g(x) = x^n - 1$. Not all of the $(2^p - 2)$ possible cyclic codes are good codes in terms of their minimum distance. We now evolve a strategy for finding good codes, i.e., of desirable minimum distance.

In the previous chapter, we learnt how to construct an extension field from the subfield. In this section we will study the prime polynomials (in a certain field) that have zeroes in the extension field. Our strategy for constructing $g(x)$ will be as follows: using the desirable zeroes in the extension field we will find prime polynomials in the subfield, multiplied together to yield a desirable $g(x)$.

Definition 5.5 A blocklength n of the form $n = q^m - 1$ is called a **Primitive Blocklength** for a code over $GF(q)$. A cyclic code over $GF(q)$ of primitive blocklength is called a **Primitive Cyclic Code**.

We have learnt that the field $GF(q^m)$ is an extension field of $GF(q)$. Let the primitive blocklength $n = q^m - 1$. Consider the factorisation

$$x^n - 1 = x^{q^m - 1} - 1 = f_1(x)f_2(x)\dots f_p(x) \quad (5.3)$$

over the field $GF(q)$. This factorisation will also be valid over the extension field $GF(q^m)$ because the addition and multiplication tables of the subfield forms a part of the tables of the extension field. We also know that $g(x)$ divides $x^n - 1$, i.e., $x^{q^m - 1} - 1$, hence $g(x)$ must be the product of some of these polynomials $f_i(x)$. Also, every non-zero element of $GF(q^m)$ is a zero of $x^{q^m - 1} - 1$. Hence, it is possible to factor $x^{q^m - 1} - 1$ in the extension field $GF(q^m)$ to get

$$x^{q^m-1} - 1 = \prod_j (x - \beta_j) \quad (5.4)$$

where β_j ranges over all the non-zero elements of $GF(q^m)$. This implies that each of the polynomials $f_i(x)$ can be represented in $GF(q^m)$ as a product of some of the linear terms, and each β_j is a zero of *exactly one* of the $f_i(x)$. This $f_i(x)$ is called the minimal polynomial of β_j .

Definition 5.6 The smallest degree polynomial with coefficients in the base field $GF(q)$ that has a zero in the extension field $GF(q^m)$ is called the **Minimal Polynomial** of β_j .

Some properties of minimal polynomials are given below.

- (i) For each $\beta \in GF(q^m)$ there exists a unique minimal polynomial, $p(x)$, with coefficients in the base field $GF(q)$.
- DID YOU KNOW ? (ii) $p(\beta) = 0$.
- (iii) $\text{Deg}(p(x)) \leq m$.
- (iv) $p(x)$ is irreducible in $GF(q)$.

Example 5.8 Consider the subfield $GF(2)$ and its extension field $GF(8)$. Here $q = 2$ and $m = 3$.

The factorisation of $x^{q^m-1} - 1$ (in the subfield/extension field) yields

$$x^{q^m-1} - 1 = x^7 - 1 = (x - 1)(x^3 + x + 1)(x^3 + x^2 + 1)$$

Next, consider the elements of the extension field $GF(8)$. The elements can be represented as $0, 1, z, z + 1, z^2, z^2 + 1, z^2 + z, z^2 + z + 1$.

Therefore, we can write

$$\begin{aligned} x^{q^m-1} - 1 &= x^7 - 1 = (x - 1)(x - z)(x - z - 1)(x - z^2)(x - z^2 - 1)(x - z^2 - z)(x - z^2 - z - 1) \\ &= (x - 1) \cdot [(x - z)(x - z^2)(x - z^2 - z)] \cdot [(x - z - 1)(x - z^2 - 1)(x - z^2 - z - 1)]. \end{aligned}$$

It can be seen that over $GF(8)$,

$$\begin{aligned} (x^3 + x + 1) &= (x - z)(x - z^2)(x - z^2 - z), \text{ and} \\ (x^3 + x^2 + 1) &= (x - z - 1)(x - z^2 - 1)(x - z^2 - z - 1). \end{aligned}$$

The multiplication and addition are carried out over $GF(8)$. Interestingly, after a little bit of algebra it is found that the coefficients of the minimal polynomial belong to $GF(2)$ only. We can now make Table 5.3.

Table 5.3 The elements of $GF(8)$ in terms of the powers of the primitive element, α

Minimal polynomial $f_i(x)$	Corresponding elements β_j in $GF(8)$	Elements in terms of powers of α
$(x - 1)$	1	α^0
$(x^3 + x + 1)$	z, z^2 and $z^2 + z$	$\alpha^1, \alpha^2, \alpha^4$
$(x^3 + x^2 + 1)$	$z + 1, z^2 + 1$ and $z^2 + z + 1$	$\alpha^3, \alpha^6, \alpha^5 (= \alpha^{12})$



Observe that the minimal polynomials are irreducible in $GF(2)$ and their degrees are less than or equal to $m = 3$. It is interesting to note the elements (in terms of powers of the primitive element α) that correspond to the same minimal polynomial. If we make the observation that $\alpha^{12} = \alpha^7 \cdot \alpha^5 = 1 \cdot \alpha^5$, we see a pattern in the elements that correspond to a certain minimal polynomial. In fact, the elements that are roots of a minimal polynomial in the extension field are of the type $\beta^{q^{r-1}}$ where β is an element of the extension field. In the above example, the zeroes of the minimal polynomial $f_2(x) = x^3 + x + 1$ are α^1, α^2 and α^4 and that of $f_3(x) = x^3 + x^2 + 1$ are α^3, α^6 and α^{12} .

We make another interesting observation. Suppose q is a prime, then an irreducible m^{th} degree polynomial, $f(x)$ with coefficients in $GF(q)$ divides $x^{q^m-1} - 1$. In this example, $q = 2$ and $m = 3$. Both the irreducible polynomials $(x^3 + x + 1)$ and $(x^3 + x^2 + 1)$ divide $x^{q^m-1} - 1 = x^7 - 1$.

Definition 5.7 Two elements of $GF(q^m)$ that share the same minimal polynomial over $GF(q)$ are called **conjugates** with respect to $GF(q)$. If $\beta \in GF(q^m)$, the conjugates of β with respect to $GF(q)$ are $\beta^{q^0}, \beta^{q^1}, \beta^{q^2}, \beta^{q^3} \dots$

Example 5.9 Let α be the primitive element of $GF(2^3)$. The elements $\{\alpha^1, \alpha^2, \alpha^4\}$ are conjugates with respect to $GF(2)$. They share the same minimal polynomial $f_2(x) = x^3 + x + 1$.

Note that

$$\alpha^{2^0} = \alpha^1$$

$$\alpha^{2^1} = \alpha^2$$

$$\alpha^{2^2} = \alpha^4$$

$$\alpha^{2^3} = \alpha^8 = \alpha^7 \alpha^1 = \alpha^1.$$

We note that this is a repetition.

$$\alpha^{2^4} = \alpha^{16} = \alpha^7 \alpha^7 \alpha^2 = \alpha^2.$$

So, we observe another repetition

As expected, there are only a finite number of conjugates in a finite field.

Example 5.10 Let α be the primitive element of $GF(2^3)$. Consider the element $\beta = \alpha^3$. To find the conjugates of β we compute the following.

$$\beta^{2^0} = \alpha^3$$

$$\beta^{2^1} = \alpha^6$$

$$\beta^{2^2} = \alpha^{12} = \alpha^7 \alpha^5 = \alpha^5$$

$$\beta^{2^3} = \alpha^{24} = \alpha^7 \alpha^7 \alpha^7 \alpha^3 = \alpha^3, \dots \text{and so forth.}$$

Thus the elements $\{\alpha^3, \alpha^6, \alpha^5\}$ are conjugates with respect to $GF(2)$. They share the same minimal polynomial $f_3(x) = x^3 + x^2 + 1$.

As we have seen, a single element in the extension field may have more than one conjugate. The conjugacy relationship between two elements depends on the base field. For example, the extension field $GF(16)$ can be

constructed using either $GF(2)$ or $GF(4)$. Two elements that are conjugates with respect to $GF(2)$ may not be conjugates with respect to $GF(4)$.

If $f(x)$ is the minimal polynomial of β , then it is also the minimal polynomial of the elements in the set $\{\beta, \beta^q, \beta^{q^2}, \dots, \beta^{q^{r-1}}\}$, where r is the smallest integer such that $\beta^{q^{r-1}} = \beta$. The set $\{\beta, \beta^q, \beta^{q^2}, \dots, \beta^{q^{r-1}}\}$ is called the **set of conjugates** or conjugacy class. The elements in the set of conjugates are all the zeroes of $f(x)$. Hence, the minimal polynomial of β can be written as

$$f(x) = (x - \beta)(x - \beta^q)(x - \beta^{q^2}) \dots (x - \beta^{q^{r-1}}) \quad (5.5)$$

Example 5.11 Consider $GF(256)$ as an extension field of $GF(2)$. Let α be the primitive element of $GF(256)$. Then a set of conjugates would be

$$\{\alpha^1, \alpha^2, \alpha^4, \alpha^8, \alpha^{16}, \alpha^{32}, \alpha^{64}, \alpha^{128}\}$$

Note that $\alpha^{256} = \alpha^{255} \alpha^1 = \alpha^1$, hence the set of conjugates terminates with α^{128} . The minimal polynomial of α is

$$f(x) = (x - \alpha^1)(x - \alpha^2)(x - \alpha^4)(x - \alpha^8)(x - \alpha^{16})(x - \alpha^{32})(x - \alpha^{64})(x - \alpha^{128})$$

The right-hand side of the equation, when multiplied, would only contain coefficients from $GF(2)$. Similarly, the minimal polynomial of α^3 would be:

$$f(x) = (x - \alpha^3)(x - \alpha^6)(x - \alpha^{12})(x - \alpha^{24})(x - \alpha^{48})(x - \alpha^{96})(x - \alpha^{192})(x - \alpha^{129})$$

Example 5.12 Let α be the primitive element of $GF(4^2)$. Here, $q = 4$. Let us find the different sets of conjugates.

Note that $\alpha^{4^0} = \alpha^1$, $\alpha^{4^1} = \alpha^4$, $\alpha^{4^2} = \alpha^{16} = \alpha^{15}\alpha^1 = \alpha^1$... and so forth.

Thus, we have the first set of conjugates, $\{\alpha^1, \alpha^4\}$.

Similarly, $(\alpha^2)^{4^0} = \alpha^2$, $(\alpha^2)^{4^1} = \alpha^8$.

Thus, we have the next set of conjugates, $\{\alpha^2, \alpha^8\}$.

Similarly, we have the other sets of conjugates, $\{\alpha^3, \alpha^{12}\}$, $\{\alpha^6, \alpha^9\}$, $\{\alpha^7, \alpha^{13}\}$, $\{\alpha^{11}, \alpha^{14}\}$, $\{\alpha^5\}$ and $\{\alpha^{10}\}$. We note that some sets have only one element.

Definition 5.8 BCH codes defined over $GF(q)$ with blocklength $q^m - 1$ are called **Primitive BCH codes**.

Having developed the necessary mathematical tools, we shall next begin our study of the BCH codes. We will develop a method for constructing the generator polynomials of BCH codes that can correct pre-specified t random errors.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 1:

1. Determine the value of α^{18} and α^{110} over $GF(16)$. Use polynomial representation.
2. Find the order of the elements $x^2 + x$ and $x^2 + x + 1$ over $GF(16)$.
3. Draw a tree showing all the sub fields of $GF(24)$ as nodes of the tree.
4. Use the primitive polynomial $p(x) = x^2 + x + 2$ over $GF(5)$ to construct the extension field $GF(5^2)$.
5. Let α be the primitive element of $GF(2^4)$. Find all the conjugates of α and the minimal polynomial.
6. Create a table of the different conjugacy classes and the corresponding minimal polynomial for $GF(4^2)$.
7. Suppose p is a prime. Show that an irreducible m^{th} degree polynomial, $f(x)$ with coefficients in $GF(p)$ divides $x^{p^m-1} - 1$.

If you have successfully solved the above problems,
you have mastered LO 1. Congratulations!



- : The answers are available here. Scan to check
- : Or
- : Visit <http://qrcode.flipick.com/index.php/602>



5.4 Generator Polynomials in Terms of Minimal Polynomials

We know that $g(x)$ is a factor of $x^n - 1$. Therefore, the generator polynomial of a cyclic code can be written in the form

$$g(x) = \text{LCM} [f_1(x)f_2(x), \dots, f_p(x)] \quad (5.6)$$

where $f_1(x), f_2(x), \dots, f_p(x)$ are the minimal polynomials of the zeroes of $g(x)$. Each minimal polynomial corresponds to a zero of $g(x)$ in an extension field. We will design good codes (i.e., determine the generator polynomials) with desirable zeroes using this approach.

Let $c(x)$ be a codeword polynomial and $e(x)$ be an error polynomial. Then the received polynomial can be written as

$$v(x) = c(x) + e(x) \quad (5.7)$$

where the polynomial coefficients are in $GF(q)$. Now consider the extension field $GF(q^m)$. Let $\gamma_1, \gamma_2, \dots, \gamma_p$ be those elements of $GF(q^m)$ which are the zeroes of $g(x)$, i.e., $g(\gamma_i) = 0$ for $i = 1, \dots, p$. Since, $c(x) = a(x)g(x)$ for some polynomial $a(x)$, we also have $c(\gamma_i) = 0$ for $i = 1, \dots, p$. Thus,

LO 2

Define BCH codes, learn how to obtain generator polynomials for BCH codes and understand how to efficiently carry out BCH decoding.

Levels of Difficulty

- S Simple:** Level 1 and Level 2 Category
- M Medium:** Level 3 and Level 4 Category
- D Difficult:** Level 5 and Level 6 Category

$$\begin{aligned} v(\gamma_i) &= c(\gamma_i) + e(\gamma_i) \\ &= e(\gamma_i) \text{ for } i = 1, \dots, p \end{aligned} \quad (5.8)$$

For a blocklength n , we have

$$v(\gamma_i) = \sum_{j=0}^{n-1} e_j \gamma_i^j \text{ for } i = 1, \dots, p \quad (5.9)$$

Thus, we have a set of p equations that involve components of the error pattern only. If it is possible to solve this set of equations for e_j , the error pattern can be precisely determined. Whether this set of equations can be solved depends on the value of p , the number of zeroes of $g(x)$. In order to solve for the error pattern, we must choose the set of p equations properly. If we have to design for a t error correcting cyclic code, our choice should be such that the set of equations can solve for at most t non-zero e_j .

Let us define the syndromes $S_i = e(\gamma_i)$ for $i = 1, \dots, p$. We wish to choose $\gamma_1, \gamma_2, \dots, \gamma_p$ in such a manner that α errors can be computed from S_1, S_2, \dots, S_p . If α is a primitive element, then the set of γ_i which allow the correction of t errors is $\{\alpha^1, \alpha^2, \alpha^3, \dots, \alpha^{2t}\}$. Thus, we have a simple mechanism of determining the generator polynomial of a BCH code that can correct t errors.



Steps for determining the generator polynomial of a t -error correcting BCH code:

For a primitive blocklength $n = q^m - 1$:

- (1) Choose a prime polynomial of degree m and construct $GF(q^m)$.
- (2) Find $f_i(x)$, the minimal polynomial of α^i for $i = 1, \dots, 2t$.
- (3) The generator polynomial for the t -error correcting code is simply

$$g(x) = \text{LCM} [f_1(x), f_2(x), \dots, f_{2t}(x)] \quad (5.10)$$

Codes designed in this manner can correct at least t errors. In many cases the codes will be able to correct more than t errors. For this reason,

$$d = 2t + 1 \quad (5.11)$$

is called the **designed distance** of the code, and the minimum distance $d^* \geq 2t + 1$. The generator polynomial has a degree equal to $n - k$ (see Theorem 4.4, previous chapter). It should be noted that once we fix n and t , we can determine the generator polynomial for the BCH code. We do not have a control over the information length k . After we determine $g(x)$, its degree will decide the value of k . Intuitively, for a fixed blocklength n , a larger value of t will force the information length k to be smaller (because a higher redundancy will be required to correct more number of errors). In the following section, we look at a few specific examples of BCH codes.

5.5 Some Examples of BCH Codes

LO 2

The following example illustrates the construction of the extension field $GF(16)$ from $GF(2)$. The minimal polynomials obtained will be used in the subsequent examples.

Example 5.13 Consider the primitive polynomial $p(z) = z^4 + z + 1$ over $GF(2)$. We shall use this to construct the extension field $GF(16)$. Let $\alpha = z$ be the primitive element. The elements of $GF(16)$ as powers of α and the corresponding minimal polynomials are listed in Table 5.4.

Table 5.4 The elements of $GF(16)$ and the corresponding minimal polynomials

Powers of α	Elements of $GF(16)$	Minimal Polynomials
α^1	z	$x^4 + x + 1$
α^2	z^2	$x^4 + x + 1$
α^3	z^3	$x^4 + x^3 + x^2 + x + 1$
α^4	$z + 1$	$x^4 + x + 1$
α^5	$z^2 + z$	$x^2 + x + 1$
α^6	$z^3 + z^2$	$x^4 + x^3 + x^2 + x + 1$
α^7	$z^3 + z + 1$	$x^4 + x^3 + 1$
α^8	$z^2 + 1$	$x^4 + x + 1$
α^9	$z^3 + z$	$x^4 + x^3 + x^2 + x + 1$
α^{10}	$z^2 + z + 1$	$x^2 + x + 1$
α^{11}	$z^3 + z^2 + z$	$x^4 + x^3 + 1$
α^{12}	$z^3 + z^2 + z + 1$	$x^4 + x^3 + x^2 + x + 1$
α^{13}	$z^3 + z^2 + 1$	$x^4 + x^3 + 1$
α^{14}	$z^3 + 1$	$x^4 + x^3 + 1$
α^{15}	1	$x + 1$

Example 5.14 We wish to determine the generator polynomial of a *single* error correcting BCH code, i.e., $t = 1$ with a blocklength $n = 15$. From (5.10), the generator polynomial for a BCH code is given by $\text{LCM}[f_1(x)f_2(x), \dots, f_{2t}(x)]$. We will make use of Table 5.4 to obtain the minimal polynomials $f_1(x)$ and $f_2(x)$. Thus, the generator polynomial of the single error correcting BCH code will be

$$\begin{aligned} g(x) &= \text{LCM}[f_1(x), f_2(x)] \\ &= \text{LCM}[(x^4 + x + 1)(x^4 + x + 1)] \\ &= x^4 + x + 1 \end{aligned}$$

Since, $\deg(g(x)) = n - k$, we have $n - k = 4$, which gives $k = 11$. Thus, we have obtained the generator polynomial of the $BCH(15, 11)$ single error correcting code. The designed distance of this code $d = 2t + 1 = 3$. It can be calculated that the minimum distance d^* of this code is also 3. Thus, in this case the designed distance is equal to the minimum distance.

Next, we wish to determine the generator polynomial of a *double* error correcting BCH code, i.e., $t = 2$ with a blocklength $n = 15$. The generator polynomial of the BCH code will be

$$\begin{aligned} g(x) &= \text{LCM}[f_1(x), f_2(x), f_3(x), f_4(x)] \\ &= \text{LCM}[(x^4 + x + 1)(x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)(x^4 + x + 1)] \\ &= (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1) \\ &= x^8 + x^7 + x^6 + x^4 + 1. \end{aligned}$$

Since, $\deg(g(x)) = n - k$, we have $n - k = 8$, which gives $k = 7$. Thus we have obtained the generator polynomial of the $BCH(15, 7)$ double error correcting code. The designed distance of this code

$d = 2t + 1 = 5$. It can be calculated that the minimum distance d^* of this code is also 5. Thus in this case the designed distance is equal to the minimum distance.

Next, we determine the generator polynomial for the *triple* error correcting binary BCH code. The generator polynomial of the BCH code will be

$$\begin{aligned} g(x) &= \text{LCM } [f_1(x)f_2(x), f_3(x), f_4(x), f_5(x), f_6(x)] \\ &= (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1) \\ &= x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1 \end{aligned}$$

In this case, $\deg(g(x)) = n - k = 10$, which gives $k = 5$. Thus we have obtained the generator polynomial of the $\text{BCH}(15, 5)$ triple error correcting code. The designed distance of this code $d = 2t + 1 = 7$. It can be calculated that the minimum distance d^* of this code is also 7. Thus, in this case, the designed distance is equal to the minimum distance.

Next, we determine the generator polynomial for a binary BCH code for the case $t = 4$. The generator polynomial of the BCH code will be

$$\begin{aligned} g(x) &= \text{LCM } [f_1(x)f_2(x), f_3(x), f_4(x), f_5(x), f_6(x)f_7(x), f_8(x)] \\ &= (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1)(x^4 + x^3 + 1) \\ &= x^{14} + x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1 \end{aligned}$$

In this case, $\deg(g(x)) = n - k = 14$, which gives $k = 1$. It can be seen that this is the simple repetition code. The designed distance of this code $d = 2t + 1 = 9$. However, it can be seen that the minimum distance d^* of this code is 15. Thus in this case the designed distance is not equal to the minimum distance, and the code is over-designed. This code can actually correct $(d^* - 1)/2 = 7$ random errors!



If we repeat the exercise for $t = 5, 6$ or 7 , we get the same generator polynomial (repetition code). Note that there are only 15 non-zero field elements in $GF(16)$ and hence there are only 15 minimal polynomials corresponding to these field elements.

Thus, we cannot go beyond $t = 7$ (because for $t = 8$ we need $f_{16}(x)$, which is undefined).

Hence, to obtain BCH codes that can correct larger number of errors we must use an extension field with more elements!

Example 5.15 We can construct $GF(16)$ as an extension field of $GF(4)$ using the primitive polynomial $p(z) = z^2 + z + 2$ over $GF(4)$. Let the elements of $GF(4)$ consist of the quaternary symbols contained in the set $\{0, 1, 2, 3\}$. The addition and multiplication tables for $GF(4)$ are given in Table 5.5 for handy reference.

Table 5.5 Addition and multiplication tables for $GF(4)$.

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

·	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

Table 5.6 lists the elements of $GF(16)$ as powers of α and the corresponding minimal polynomials.

Table 5.6 The elements of $GF(16)$ as powers of α

Powers of α	Elements of $GF(16)$	Minimal Polynomials
α^1	z	$x^2 + x + 2$
α^2	$z + 2$	$x^2 + x + 3$
α^3	$3z + 2$	$x^2 + 3x + 1$
α^4	$z + 1$	$x^2 + x + 2$
α^5	2	$x + 2$
α^6	$2z$	$x^2 + 2x + 1$
α^7	$2z + 3$	$x^2 + 2x + 2$
α^8	$z + 3$	$x^2 + x + 3$
α^9	$2z + 2$	$x^2 + 2x + 1$
α^{10}	3	$x + 3$
α^{11}	$3z$	$x^2 + 3x + 3$
α^{12}	$3z + 1$	$x^2 + 3x + 1$
α^{13}	$2z + 1$	$x^2 + 2x + 2$
α^{14}	$3z + 3$	$x^2 + 3x + 3$
α^{15}	1	$x + 1$

For $t = 1$,

$$\begin{aligned} g(x) &= \text{LCM } [f_1(x), f_2(x)] \\ &= \text{LCM}[(x^2 + x + 2)(x^2 + x + 3)] \\ &= x^4 + x + 1 \end{aligned}$$

Since, $\deg(g(x)) = n - k$, we have $n - k = 4$, which gives $k = 11$. Thus we have obtained the generator polynomial of the *single* error correcting BCH(15, 11) code over $GF(4)$. It takes in 11 quaternary information symbols and encodes them into 15 quaternary symbols. Note that one quaternary symbol is equivalent to two bits. So, in effect, the BCH(15, 11) takes in 22 input bits and transforms them into 30 encoded bits (can this code be used to correct a burst of length 2 for a binary sequence of length 30?). The designed distance of this code $d = 2t + 1 = 3$. It can be calculated that the minimum distance d^* of this code is also 3. Thus in this case the designed distance is equal to the minimum distance.

For $t = 2$,

$$\begin{aligned} g(x) &= \text{LCM } [f_1(x), f_2(x), f_3(x), f_4(x)] \\ &= \text{LCM}[(x^2 + x + 2)(x^2 + x + 3)(x^2 + 3x + 1)(x^2 + x + 2)] \\ &= (x^2 + x + 2)(x^2 + x + 3)(x^2 + 3x + 1) \\ &= x^6 + 3x^5 + x^4 + x^3 + 2x^2 + 2x + 1 \end{aligned}$$

This is the generator polynomial of a (15, 9) *double* error correcting BCH code over $GF(4)$.

For $t = 3$,

$$\begin{aligned} g(x) &= \text{LCM } [f_1(x), f_2(x), f_3(x), f_4(x), f_5(x), f_6(x)] \\ &= \text{LCM}[(x^2 + x + 2)(x^2 + x + 3)(x^2 + 3x + 1)(x^2 + x + 2)(x + 2)(x^2 + 2x + 1)] \end{aligned}$$

$$\begin{aligned}
 &= (x^2 + x + 2)(x^2 + x + 3)(x^2 + 3x + 1)(x + 2)(x^2 + 2x + 1) \\
 &= x^9 + 3x^8 + 3x^7 + 2x^6 + x^5 + 2x^4 + x + 2
 \end{aligned}$$

This is the generator polynomial of a (15, 6) *triple* error correcting BCH code over $GF(4)$.

Similarly, for $t = 4$,

$$g(x) = x^{11} + x^{10} + 2x^8 + 3x^7 + 3x^6 + x^5 + 3x^4 + x^3 + x + 3$$

This is the generator polynomial of a (15, 4) *four* error correcting BCH code over $GF(4)$.

Similarly, for $t = 5$,

$$g(x) = x^{12} + 2x^{11} + 3x^{10} + 2x^9 + 2x^8 + x^7 + 3x^6 + 3x^4 + 3x^3 + x^2 + 2$$

This is the generator polynomial of a (15, 3) *five* error correcting BCH code over $GF(4)$.

Similarly, for $t = 6$,

$$g(x) = x^{14} + x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$$

This is the generator polynomial of a (15, 1) *six* error correcting BCH code over $GF(4)$. As is obvious, this is the simple repetition code, and can correct up to seven errors.

Table 5.7 lists the generator polynomials of binary BCH codes of length up to $2^5 - 1$. Suppose we wish to construct generator polynomials of the $BCH(15,7)$ code then, from the table we have (111 010 001) for the coefficients of the generator polynomial. Therefore, $g(x) = x^8 + x^7 + x^6 + x^4 + 1$.

Table 5.7 The generator polynomials of binary BCH codes of length up to $2^5 - 1$

<i>n</i>	<i>k</i>	<i>t</i>	Generator Polynomial Coefficients
7	4	1	1 011
15	11	1	10 011
15	7	2	111 010 001
15	5	3	10 100 110 111
31	26	1	100 101
31	21	2	11 101 101 001
31	16	3	1 000 111 110 101 111
31	11	5	101 100 010 011 011 010 101
31	6	7	11 001 011 011 110 101 000 100 111

5.6 Decoding of BCH Codes

LO 2

So far we have learnt to obtain the generator polynomial for a BCH code given the number of random errors to be corrected. With the knowledge of the generator polynomial, very fast encoders can be built in hardware. We now shift our attention to the decoding of the BCH codes. Since the BCH codes are a subclass of the cyclic codes, any standard decoding procedure for cyclic codes is also applicable to BCH codes. However, better, more efficient algorithms have been designed specifically for BCH codes. We discuss the **Gorenstein-**

Zierler decoding algorithm, which is the generalised form of the binary decoding algorithm first proposed by Petersen.

We develop here the decoding algorithm for a t -error correcting BCH code. Suppose a BCH code is constructed based on the field element α then, considering the error polynomial

$$e(x) = e_{n-1}x^{n-1} + e_{n-2}x^{n-2} + \dots + e_1x + e_0 \quad (5.12)$$

where, at the most, t coefficients are non-zero. Again, suppose that v errors actually occur, where $0 \leq v \leq t$, and that these errors occur at locations i_1, i_2, \dots, i_v , then error polynomial can be written as

$$e(x) = e_{i_1}x^{i_1} + e_{i_2}x^{i_2} + \dots + e_{i_v}x^{i_v} \quad (5.13)$$

where e_{i_k} is the magnitude of the k^{th} error. Note that we are considering the general case. For binary codes, $e_{i_k} = 1$. For error correction, we must know two things:

- (i) where the errors have occurred, i.e. the error locations, and
- (ii) what are the *magnitudes* of these errors.

Thus, the unknowns are i_1, i_2, \dots, i_v and $e_{i_1}, e_{i_2}, \dots, e_{i_v}$, which signify the locations and the magnitudes of the errors respectively. The syndrome can be obtained by evaluating the received polynomial at α .

$$\begin{aligned} S_1 &= v(\alpha) = c(\alpha) + e(\alpha) = e(\alpha) \\ &= e_{i_1}x^{i_1} + e_{i_2}x^{i_2} + \dots + e_{i_v}x^{i_v} \end{aligned} \quad (5.14)$$

Next, define the error magnitudes, $Y_k = e_{i_k}$ for $k = 1, 2, \dots, v$ and the error locations $X_k = \alpha^{i_k}$ for $k = 1, 2, \dots, v$, where i_k is the location of the k^{th} error and X_k is the field element associated with this location. Now, the syndrome can be written as

$$S_1 = Y_1X_1 + Y_2X_2 + \dots + Y_vX_v \quad (5.15)$$

We can evaluate the received polynomial at each of the powers of α that have been used to define $g(x)$. We define the syndromes for $j = 1, 2, \dots, 2t$ by

$$S_j = v(\alpha^j) = c(\alpha^j) + e(\alpha^j) = e(\alpha^j) \quad (5.16)$$

Thus, we have the following set of $2t$ simultaneous equations, with v unknown error locations X_1, X_2, \dots, X_v and the v unknown error magnitudes Y_1, Y_2, \dots, Y_v .

$$\begin{aligned} S_1 &= Y_1X_1 + Y_2X_2 + \dots + Y_vX_v \\ S_2 &= Y_1X_1^2 + Y_2X_2^2 + \dots + Y_vX_v^2 \\ &\vdots && \vdots \\ S_{2t} &= Y_1X_1^{2t} + Y_2X_2^{2t} + \dots + Y_vX_v^{2t} \end{aligned} \quad (5.17)$$

Next, define the **error locator polynomial**

$$\Lambda(x) = \Lambda_vx^v + \Lambda_{v-1}x^{v-1} + \dots + \Lambda_1x + 1 \quad (5.18)$$

The zeroes of this polynomial are the inverse error locations X_k^{-1} for $k = 1, 2, \dots, v$. That is,

$$\Lambda(x) = (1 - xX_1)(1 - xX_2)\dots(1 - xX_v) \quad (5.19)$$

So, if we know the coefficients of the error locator polynomial $\Lambda(x)$, we can obtain the error locations X_1, X_2, \dots, X_v . After some algebraic manipulations we obtain

$$\Lambda_1 S_{j+v-1} + \Lambda_2 S_{j+v-2} + \dots + \Lambda_v S_j = -S_{j+v} \text{ for } j = 1, 2, \dots, v \quad (5.20)$$

This is nothing but a set of linear equations that relate the syndromes to the coefficients of $\Lambda(x)$. This set of equations can be written in the matrix form as follows:

$$\begin{bmatrix} S_1 & S_2 & \dots & S_{v-1} & S_v \\ S_2 & S_3 & \dots & S_v & S_{v+1} \\ S_v & S_{v+1} & \dots & S_{2v-2} & S_{2v-1} \end{bmatrix} \begin{bmatrix} \Lambda_v \\ \Lambda_{v-1} \\ \vdots \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} -S_{v+1} \\ -S_{v+2} \\ \vdots \\ -S_{2v} \end{bmatrix} \quad (5.21)$$

The values of the coefficients of the error locator polynomial can be determined by inverting the syndrome matrix. This is possible only if the matrix is non-singular. It can be shown that this matrix is non-singular if there are v errors.

DID YOU KNOW ? Steps for decoding BCH codes

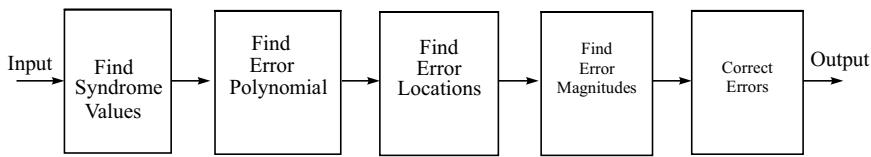
- (1) As a trial value, set $v = t$ and compute the determinant of the matrix of syndromes, M . If the determinant is zero, set $v = t - 1$. Again compute the determinant of M . Repeat this process until a value of v is found for which the determinant of the matrix of syndromes is non-zero. This value of v is the actual number of errors that occurred.
- (2) Invert the matrix M and find the coefficients of the error locator polynomial $\Lambda(x)$.
- (3) Solve $\Lambda(x) = 0$ to obtain the zeroes and from them compute the error locations X_1, X_2, \dots, X_v . If it is a binary code, stop (because the magnitudes of error are unity).
- (4) If the code is not binary, go back to the system of equations:

$$\begin{aligned} S_1 &= Y_1 X_1 + Y_2 X_2 + \dots + Y_v X_v \\ S_2 &= Y_1 X_1^2 + Y_2 X_2^2 + \dots + Y_v X_v^2 \\ &\vdots && \vdots \\ S_{2t} &= Y_1 X_1^{2t} + Y_2 X_2^{2t} + \dots + Y_v X_v^{2t} \end{aligned}$$

Since the error locations are now known, these form a set of $2t$ linear equations. These can be solved to obtain the error magnitudes.

Solving for the Λ_i by inverting the $v \times v$ matrix can be computationally expensive. The number of computations required will be proportional to v^3 . If we need to correct a large number of errors (i.e., a large v) we need more efficient ways to solve the matrix equation. Various refinements have been found which greatly reduce the computational complexity. It can be seen that the $v \times v$ matrix is not arbitrary in form. The entries in its diagonal perpendicular to the main diagonal are all identical. This property is called *persymmetry*. This structure was exploited by Berleykamp (1968) and Massey (1969) to solve the system of equations in a much simpler manner. An algorithm proposed by Forney is used for finding the error magnitudes. The **Berleykamp-Massey-Forney** algorithm is commonly used for decoding BCH codes.

The simplest way to search for the zeroes of $\Lambda(x)$ is to test out all the field elements one by one. This method of exhaustive search is known as the *Chien search*. The entire flow of the BCH decoding is shown in Fig. 5.1.

**Fig. 5.1** The flow for BCH decoding.

Example 5.16 Consider the $\text{BCH}(15, 5)$ triple error correcting code with the generator polynomial

$$g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$$

Let the all-zero codewords be transmitted and the received polynomial be $v(x) = x^5 + x^3$. Thus there are two errors at the locations 4 and 10. The error polynomial $e(x) = x^5 + x^3$. But the decoder does not know this. It does not even know how many errors have actually occurred. We use the Gorenstein-Zierler decoding algorithm. First, we compute the syndromes using the arithmetic of $GF(16)$.

$$\begin{aligned} S_1 &= \alpha^5 + \alpha^3 = \alpha^{11} \\ S_2 &= \alpha^{10} + \alpha^6 = \alpha^7 \\ S_3 &= \alpha^{15} + \alpha^9 = \alpha^7 \\ S_4 &= \alpha^{20} + \alpha^{12} = \alpha^{14} \\ S_5 &= \alpha^{25} + \alpha^{15} = \alpha^5 \\ S_6 &= \alpha^{30} + \alpha^{18} = \alpha^{14} \end{aligned}$$

First, set $v = t = 3$, since this is a triple error correcting code.

$$\mathbf{M} = \begin{bmatrix} S_1 & S_2 & S_3 \\ S_2 & S_3 & S_4 \\ S_3 & S_4 & S_5 \end{bmatrix} = \begin{bmatrix} \alpha^{11} & \alpha^7 & \alpha^7 \\ \alpha^7 & \alpha^7 & \alpha^{14} \\ \alpha^7 & \alpha^{14} & \alpha^5 \end{bmatrix}$$

$\text{Det}(\mathbf{M}) = 0$, which implies that fewer than 3 errors have occurred.

Next, set $v = t = 2$.

$$\mathbf{M} = \begin{bmatrix} S_1 & S_2 \\ S_2 & S_3 \end{bmatrix} = \begin{bmatrix} \alpha^{11} & \alpha^7 \\ \alpha^7 & \alpha^7 \end{bmatrix}$$

$\text{Det}(\mathbf{M}) = 1$, which implies that 2 errors have actually occurred.

We next calculate \mathbf{M}^{-1} . In this case,

$$\begin{aligned} \mathbf{M}^{-1} &= \begin{bmatrix} \alpha^7 & \alpha^7 \\ \alpha^7 & \alpha^{11} \end{bmatrix} \\ \begin{bmatrix} \Lambda_2 \\ \Lambda_1 \end{bmatrix} &= M^{-1} \begin{bmatrix} -S_3 \\ -S_4 \end{bmatrix} = \begin{bmatrix} \alpha^7 & \alpha^7 \\ \alpha^7 & \alpha^{11} \end{bmatrix} \begin{bmatrix} \alpha^7 \\ \alpha^{14} \end{bmatrix} \end{aligned}$$

Solving for Λ_1 and Λ_2 we get $\Lambda_2 = \alpha^8$ and $\Lambda_1 = \alpha^{11}$. Thus,

$$\Lambda(x) = \alpha^8 x^2 + \alpha^{11}x + 1 = (\alpha^5x + 1)(\alpha^3x + 1)$$

Thus, the recovered error locations are α^5 and α^3 . Since the code is binary, the error magnitudes are 1. Thus, $e(x) = x^5 + x^3$.

Next, suppose the received vector was $v(x) = x^7 + x^2$ corresponding to an all-zero vector being transmitted. The steps are similar. We must recalculate the syndromes.

$$\begin{aligned} S_1 &= \alpha^7 + \alpha^2 = \alpha^{12} \\ S_2 &= \alpha^{14} + \alpha^4 = \alpha^9 \\ S_3 &= \alpha^{21} + \alpha^6 = 0 \\ S_4 &= \alpha^{28} + \alpha^8 = \alpha^3 \\ S_5 &= \alpha^{35} + \alpha^{10} = 1 \\ S_6 &= \alpha^{42} + \alpha^{12} = 0 \end{aligned}$$

First, set $v = t = 3$, since this is a triple error correcting code.

$$\mathbf{M} = \begin{bmatrix} S_1 & S_2 & S_3 \\ S_2 & S_3 & S_4 \\ S_3 & S_4 & S_5 \end{bmatrix} = \begin{bmatrix} \alpha^{12} & \alpha^0 & 0 \\ \alpha^9 & 0 & \alpha^3 \\ 0 & \alpha^3 & 1 \end{bmatrix}$$

$\text{Det}(\mathbf{M}) = 0$, which implies that fewer than 3 errors have occurred.

Next, set $v = t = 2$.

$$\mathbf{M} = \begin{bmatrix} S_1 & S_2 \\ S_2 & S_3 \end{bmatrix} = \begin{bmatrix} \alpha^{12} & \alpha^9 \\ \alpha^9 & 0 \end{bmatrix}$$

$\text{Det}(\mathbf{M}) \neq 0$, which implies that 2 errors have actually occurred.

We next calculate \mathbf{M}^{-1} . In this case,

$$\begin{aligned} \mathbf{M}^{-1} &= \begin{bmatrix} 0 & \alpha^6 \\ \alpha^6 & \alpha^9 \end{bmatrix} \\ \begin{bmatrix} \Lambda_2 \\ \Lambda_1 \end{bmatrix} &= M^{-1} \begin{bmatrix} -S_3 \\ -S_4 \end{bmatrix} = \begin{bmatrix} 0 & \alpha^6 \\ \alpha^6 & \alpha^9 \end{bmatrix} \begin{bmatrix} 0 \\ \alpha^3 \end{bmatrix} \end{aligned}$$

Solving for Λ_1 and Λ_2 we get $\Lambda_2 = \alpha^9$ and $\Lambda_1 = \alpha^{12}$. Thus,

$$\Lambda(x) = \alpha^9 x^2 + \alpha^{12}x + 1 = (\alpha^7x + 1)(\alpha^2x + 1).$$

Thus, the recovered error locations are α^7 and α^2 . Since the code is binary, the error magnitudes are 1. Thus, $e(x) = x^7 + x^2$.

In the next section, we will study the famous Reed-Solomon codes, an important sub-class of BCH codes.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 2:

1. Let the primitive blocklength $n = 31$ with $q = 2$, and $m = 5$. Determine the generator polynomial for a single error correcting BCH code. M
2. For a binary BCH code with $n = 31$ find the generator polynomial for a triple error correcting code. M
3. Consider a quaternary BCH code with $n = 255$. Find the generator polynomial for a single error correcting code. M
4. Consider a double error correcting binary BCH code with $n = 31$. Create a table of the weight of the codewords and the number of such codewords with that weight. D

If you have successfully solved the above problems,
you have mastered LO 2. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/603>



5.7 Reed-Solomon Codes

Reed-Solomon (RS) codes are an important subset of the non-binary BCH codes with a wide range of applications in digital communications and data storage. The typical application areas of the RS code are:

- Storage devices (including tapes, Compact Disks, DVDs, barcodes, etc.),
- Wireless or mobile communications (including cellular telephones, microwave links, etc.),
- Satellite communications,
- Digital television / Digital Video Broadcast (DVB),
- High-speed modems such as those employing ADSL, xDSL, etc.

LO 3



Explain the powerful Reed-Solomon (RS) codes and understand how RS encoders and decoders work.

It all began with a five-page paper that appeared in 1960 in the *Journal of the Society for Industrial and Applied Mathematics*. The paper, “Polynomial Codes over Certain Finite Fields” by Irving S. Reed and Gustave Solomon, then staff members at MIT’s Lincoln Laboratory, introduced the ideas that form a significant portion of current error correcting techniques for everything from computer hard disk drives to CD players. Reed-Solomon codes (plus a lot of engineering wizardry, of course) made possible the stunning pictures of the outer planets sent back by Voyager II. RS codes were used by NASA for data transmission from Skylab, International Space Station, the Hubble space telescope and the Mars mission in 2001. They make it possible to scratch a compact disc (CD) and still enjoy the music. And in the not-too-distant future,

they will enable the profit mongers of cable television to squeeze more than 700 channels into their systems. The DSC-HDTV uses RS code with $t = 10$.

There is a payoff for a coding system based on groups of bits, such as bytes, rather than individual 0s and 1s. That feature makes Reed-Solomon codes particularly good at dealing with *bursts* of errors: six consecutive bit errors, for example, can affect at most two bytes. Thus, even a double-error-correction version of a Reed-Solomon code can provide a comfortable safety factor. Current implementations of Reed-Solomon codes in CD technology are able to cope with error bursts as long as 4000 consecutive bits.

In this sub-class of BCH codes, the symbol field $GF(q)$ and the error locator field $GF(q^m)$ are the same, i.e., $m = 1$. Thus, in this case

$$n = q^m - 1 = q - 1 \quad (5.22)$$

The minimal polynomial of any element β in the same field $GF(q)$ is

$$f_\beta(x) = x - \beta \quad (5.23)$$



Since the symbol field (sub-field) and the error locator field (extension field) are the same, all the minimal polynomials are linear. The generator polynomial for a t -error correcting code will simply be:

$$\begin{aligned} g(x) &= \text{LCM}[f_1(x)f_2(x), \dots, f_{2t}(x)] \\ &= (x - \alpha)(x - \alpha^2) \dots (x - \alpha^{2t-1})(x - \alpha^{2t}) \end{aligned} \quad (5.24)$$

Hence, the degree of the generator polynomial will always be $2t$. Thus, the RS code satisfies

$$n - k = 2t \quad (5.25)$$

In general, the generator polynomial of an RS code can be written as

$$g(x) = (x - \alpha^i)(x - \alpha^{i+1}) \dots (x - \alpha^{2t+i-1})(x - \alpha^{2t+i}) \quad (5.26)$$

Example 5.17 Consider the double error correcting RS code of blocklength 7 over $GF(2^3)$. Here $t = 2$. We use here the elements of the extension field $GF(8)$ constructed from $GF(2)$ using the primitive polynomial $p(z) = z^3 + z + 1$. The generator polynomial can be written as:

$$\begin{aligned} g(x) &= (x - \alpha)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4) \\ &= x^4 + \alpha^3 x^3 + x^2 + \alpha x + \alpha^3 \end{aligned}$$

Here $n - k = 4$, which implies $k = 3$. Thus, we have obtained the generator polynomial of an RS $(7, 3)$ code over $GF(8)$. Each symbol in $GF(8)$ can be represented using 3 bits. Note that this coding procedure takes in 3 symbols (equivalent to $3 \times 3 = 9$ bits) and encodes them into 7 symbols (equivalent to 21 bits). The total number of codewords in this RS $(7, 3)$ code is $q^k = 8^3$ codewords.

Example 5.18 Consider the double error correcting RS code of blocklength 15 over $GF(16)$.

Here $t = 2$. We use here the elements of the extension field $GF(16)$ constructed from $GF(2)$ using the primitive polynomial $p(z) = z^4 + z + 1$. The generator polynomial can be written as

$$\begin{aligned} g(x) &= (x - \alpha)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4) \\ &= x^4 + (z^3 + z^2 + 1)x^3 + (z^3 + z^2)x^2 + z^3x + (z^2 + z + 1) \\ &= x^4 + (\alpha^{13})x^3 + (\alpha^6)x^2 + (\alpha^3)x + \alpha^{10} \end{aligned}$$

Here $n - k = 4$, which implies $k = 11$. Thus we have obtained the generator polynomial of an RS(15, 11) code over $GF(16)$. Note that this coding procedure takes in 11 symbols (equivalent to $4 \times 11 = 44$ bits) and encodes them into 15 symbols (equivalent to 60 bits).

Example 5.19 Consider the double error correcting RS code of blocklength 7 over $GF(2^3)$ with the generator polynomial $g(x) = x^4 + \alpha^3x^3 + x^2 + \alpha x + \alpha^3$. Suppose we have to encode the information polynomial given by $i(x) = \alpha^6x^2 + \alpha^2x$ (this corresponds to the sequence: 0, α^2 , α^6). The corresponding codeword polynomial will be

$$\begin{aligned} c(x) &= i(x)g(x) = (\alpha^6x^2 + \alpha^2x)(x^4 + \alpha^3x^3 + x^2 + \alpha x + \alpha^3) \\ &= \alpha^6x^6 + \alpha x^4 + \alpha^6x^3 + \alpha^5x^2 + \alpha^5x \\ &= \alpha^6x^6 + 0x^5 + \alpha x^4 + \alpha^6x^3 + \alpha^5x^2 + \alpha^5x + 0 \end{aligned}$$

The codeword will be $[0 \ \alpha^5 \ \alpha^5 \ \alpha^6 \ \alpha \ 0 \ \alpha^6]$.

Theorem 5.2 A Reed-Solomon code is a maximum distance separable (MDS) code and its minimum distance is $n - k + 1$.

Proof: Let the designed distance of the RS code be $d = 2t + 1$. The minimum distance d^* satisfies the condition $d^* \geq d = 2t + 1$.

But, for an RS code, $2t = n - k$. Hence,

$$d^* \geq d = 2t + 1 = n - k + 1$$

But, by the Singleton bound for any linear code,

$$d^* \leq n - k + 1$$

Thus, $d^* = n - k + 1$, and the minimum distance $d^* = d$, the designed distance of the code.

 **Intuition** Since RS codes are maximum distance separable (MDS), all of the possible codewords are as far away as possible algebraically in the code space. It implies a uniform codeword distribution in the code space.

Table 5.8 lists the parameters of some RS codes. Note that for a given minimum distance, in order to have a high code rate, one must work with larger Galois Fields.

Table 5.8 Some RS code parameters

M	$q = 2^m$	$n = q - 1$	t	k	d^*	$r = k/n$
3	8	7	1	1	3	0.3333
			1	5	3	0.7143
			2	3	5	0.4286
			3	1	7	0.1429
			1	13	3	0.8667
			2	11	5	0.7333
4	16	15	3	9	7	0.6000
			4	7	9	0.4667
			5	5	11	0.3333
			6	3	13	0.2000
			7	1	15	0.0667
						<i>Contd.</i>

			1	29	3	0.9355
			5	21	11	0.6774
5	32	31	8	15	17	0.4839
			5	245	11	0.9608
			15	225	31	0.8824
8	256	255	50	155	101	0.6078

Example 5.20 A popular Reed-Solomon code is RS (255, 223) with 8-bit symbols (bytes), i.e., over $GF(255)$. Each codeword contains 255 codeword bytes, of which 223 bytes are data and 32 bytes are parity. For this code, $n = 255$, $k = 223$ and $n - k = 32$. Hence, $2t = 32$, or $t = 16$. Thus, the decoder can correct any 16 symbol random error in the codeword, i.e. errors in up to 16 bytes anywhere in the codeword can be corrected. Suppose these errors in the codeword are contiguous. The RS (255, 223) can correct 16 contiguous symbols in error. Thus, it can correct a burst of $16 \times 8 = 128$ bits.

Example 5.21 Reed-Solomon error correction codes can have an extremely pronounced effect on the efficiency of a digital communication channel. For example, an operation running at a data rate of 1 million bytes per second will carry approximately 4000 blocks of 255 bytes each second. If 1000 random short errors (less than 17 bits in length) per second are injected into the channel, about 600 to 800 blocks per second would be corrupted, which might require retransmission of nearly all of the blocks. By applying the Reed-Solomon (255, 235) code (that corrects up to 10 errors per block of 235 information bytes and 20 parity bytes), the typical time between blocks that cannot be corrected and would require retransmission will be about 800 years. The meantime between incorrectly decoded blocks will be over 20 billion years!

DID YOU KNOW An (n, k) RS code can be punctured by deleting any one of its symbols to obtain a $(n - 1, k)$ code. An (n, k) RS code can be extended by adding parity check symbols. A single-extended RS code is also an MDS code.

5.8 Implementation of Reed-Solomon Encoders and Decoders

LO 3

Explain the powerful Reed-Solomon (RS) codes and understand how RS encoders and decoders work.

Hardware Implementation Let the generator polynomial of the RS encoder be represented by

$$g(x) = g_0 + g_1x + g_2x^2 + \dots + g_{2t-1}x^{2t-1} + x^{2t} \quad (5.27)$$

Note that the right hand side of the equation is written such that the highest power of x is to the right. The hardware implementation of this RS encoder is shown in Fig. 5.2. The encoding steps are as follows:

1. Switch 1 is closed during the first k clock cycles to allow shifting the message symbols into the $(n - k)$ stage shift-register. Note that in these k cycles, the contents of the shift-register and the feedback loop is continually changing, based on the information symbols being shifted in, as well as, the additions prior to each element of the shift-register.

2. Switch 2 is in the down position during the first k clock cycles in order to allow simultaneous transfer of the message symbols directly to an output register. Note that this is a systematic design, and the first k symbols of the codeword are the original information symbols.
3. After transfer of the k^{th} message symbol to the output register, switch 1 is opened and switch 2 is moved to the up position. What is residing in the shift-register elements are the parity symbols. These symbols are ready to be shifted out, and appended to the information symbols in order to create the entire codeword.
4. During the remaining $(n - k)$ clock cycles, the parity symbols contained in the shift-register need to be cleared by moving them to the output register.
5. The total number of clock cycles is equal to n , and the contents of the output register are the final codeword polynomial, corresponding to the k information symbols.

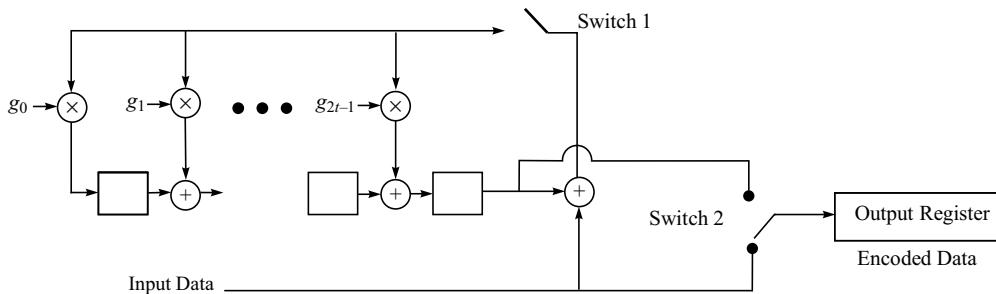


Fig. 5.2 The hardware implementation of a systematic RS encoder with a generator polynomial $g(x) = g_0 + g_1x + g_2x^2 + \dots + g_{2t-1}x^{2t-1} + x^{2t}$.

Example 5.22 The IEEE 802.15.4a standard uses an RS (63, 55) encoder, implemented over $GF(2^6)$. Here, $n - k = 63 - 55 = 8 = 2t$. The generator polynomial of the encoder can be written as

$$\begin{aligned} g(x) &= (x - \alpha)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4)(x - \alpha^5)(x - \alpha^6)(x - \alpha^7)(x - \alpha^8) \\ &= x^8 + \alpha^{43}x^7 + \alpha^{59}x^6 + \alpha^{31}x^5 + \alpha^{10}x^4 + \alpha^{40}x^3 + \alpha^{14}x^2 + \alpha^7x + \alpha^{36} \end{aligned} \quad (5.28)$$

The shift-register portion of the encoder is shown in Fig. 5.3.

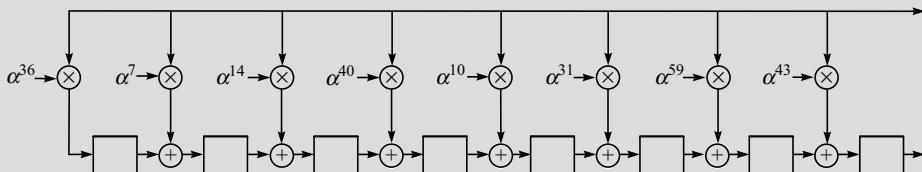


Fig. 5.3 The shift-register portion of the encoder for RS(63, 55).

The decoding of RS codes follows the procedure depicted in Fig. 5.1. Each of the sub-blocks, i.e., the syndrome calculator block, the block for error location and the block for error magnitudes can be implemented using an FPGA. Since all the arithmetic is done over $GF(q)$, lookup tables are used for addition, multiplication and inversion.

A number of commercial hardware implementations exist for the hardware implementation of RS codes. Many existing systems use *off-the-shelf* integrated circuits that encode and decode Reed-Solomon codes.

These ICs tend to support a certain amount of programmability, for example, RS(255, k) where $t = 1$ to 16 symbols. A recent trend is towards VHDL or Verilog designs (logic cores or intellectual property cores). These have a number of advantages over standard ICs. A logic core can be integrated with other VHDL or Verilog components and synthesised to an FPGA (Field Programmable Gate Array) or ASIC (Application Specific Integrated Circuit) – this enables so-called “System on Chip” designs where multiple modules can be combined in a single IC. Depending on production volumes, logic cores can often give significantly lower system costs than standard ICs. By using logic cores, a designer avoids the potential need to buy a Reed-Solomon IC for a lifetime.

Software Implementation Until recently, software implementations in “real-time” required too much computational power for all but the simplest of Reed-Solomon codes (i.e. codes with small values of t). The major difficulty in implementing Reed-Solomon codes in software is that general purpose processors do not support Galois field arithmetic operations. For example, to implement a Galois field multiply in software requires a test for 0, two log table look-ups, modulo add and anti-log table look-ups. However, careful design, together with increase in processor performance, means that software implementations can operate at relatively high data rates. Table 5.9 gives sample benchmark figures on a 1.6 GHz Pentium PC. These data rates are for decoding only. Encoding is considerably faster since it requires less computation.

Table 5.9 Sample benchmark figures for software decoding of some RS codes

Code	Data Rate	t
RS(255, 251)	~ 120 Mbps	2
RS(255, 239)	~ 30 Mbps	8
RS(255, 223)	~ 10 Mbps	16

5.9 Performance of RS Codes Over Real Channels

LO 3



One may be tempted to believe that as we decrease the code rate of an RS code, the BER performance would improve monotonically. However, in real world communications, the modulation scheme also plays a role with respect to the BER (Bit Error Rate). Thus, both modulation and coding mechanisms have to be considered. One mechanism works to improve error performance while the other works to degrade it. The improving mechanism is the coding. The greater the redundancy, the greater will be the error-correcting capability of the code. The degrading mechanism is the energy reduction per channel symbol (compared to the data symbol) that results from the increased redundancy (and faster signaling in a real-time communication system). The reduced symbol energy causes the demodulator to make more errors. Eventually, the second mechanism wins out, and thus at very low code rates the system experiences error-performance degradation.

Figure 5.4 shows the performance of a real-time communication system considering both the error control coding and modulation. An RS (31, k) code together with a BPSK modulation scheme has been considered. Here, the price paid for error-correction coding is bandwidth expansion by a factor equal to the inverse of the code rate. The curves show clear optimum code rates that minimise the required E_b/N_0 . We see that the optimum code rate is about 0.6 to 0.7 for a Gaussian channel, 0.5 for a Rician-fading channel (with the ratio of direct to reflected received signal power, $K = 7$ dB), and 0.3 for a Rayleigh-fading channel.

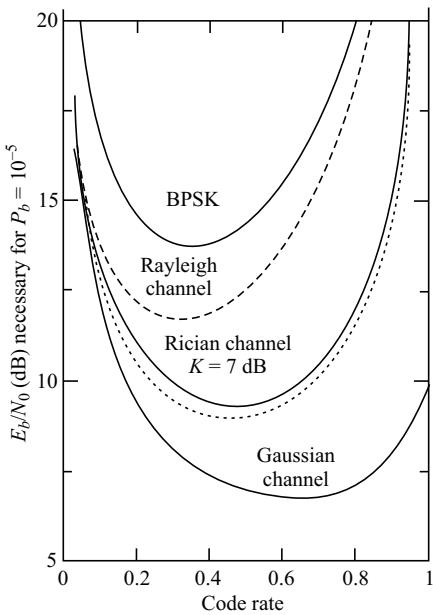


Fig. 5.4 BPSK plus RS (31, k) decoder performance as a function of code rate.

Energy consumption by the communication system is also becoming an important design constraint in many real-world scenarios, for example in the wireless sensor networks. If we consider the hardware implementation of the error control coding block and the modulator block, the energy expended can be categorised as follows:

- (i) Computational energy for channel coding and decoding,
- (ii) Circuit energy for the modulator and demodulator,
- (iii) Signal energy (radio energy) for transmitting the redundant bits.

At short distances, the energy consumed in transceiver circuitry and computation is comparable to the signal energy. Thus, the design strategy is to minimise the overall energy consumption in the encoder-decoder blocks and the modulator-demodulator blocks, taking into consideration the transmitted signal energy as well. Error control codes used to achieve the desired bit error rate, reduces the signal energy due to their coding gain. However, more energy is spent for transmitting the redundant bits and for encoding-decoding. Therefore, for a particular coding-modulation configuration, energy consumption is optimal. This optimal pair will also depend on the distance between the transmitter and receiver, because that will have an impact on the signal (radio) energy.

Example 5.23 Figure 5.5 shows the energy required to obtain $\text{BER} = 10^{-5}$ for different RS(31, k) coders and BPSK modulation. The total energy per data bit is plotted against the error correcting capability, t . The code is run on a StrongArm SA1100 processor, with a transmitter receiver separation of 110 m and the path loss exponent equal to 4. The total computation energy is the sum of encoding and decoding energy. It can be seen from the figure that the RS(31, 29) minimises the overall energy for this processor.

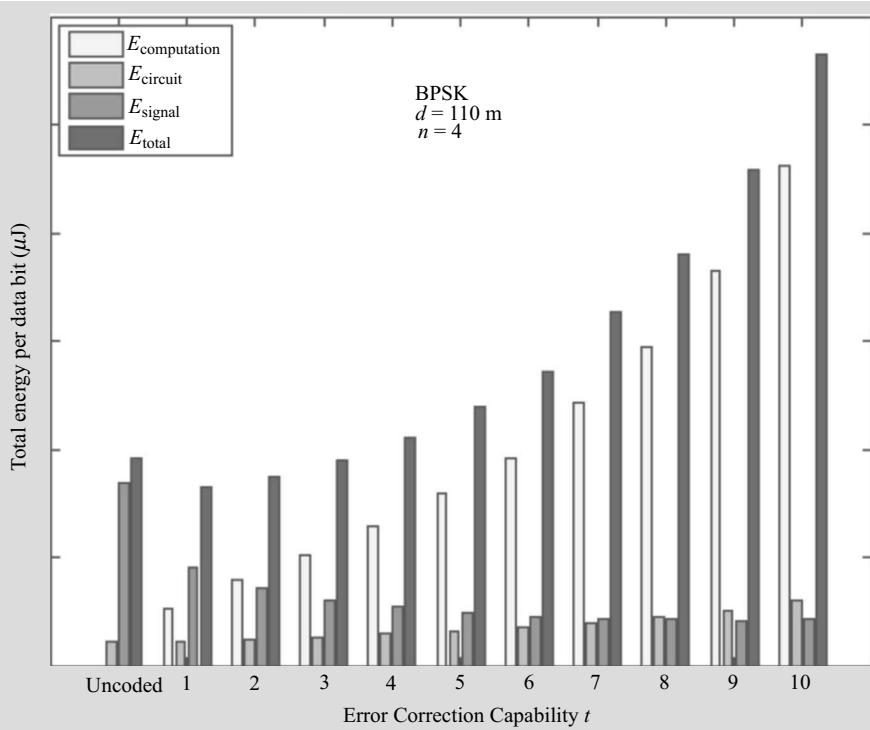


Fig. 5.5 Total energy versus the error correcting capability of RS(31, k).

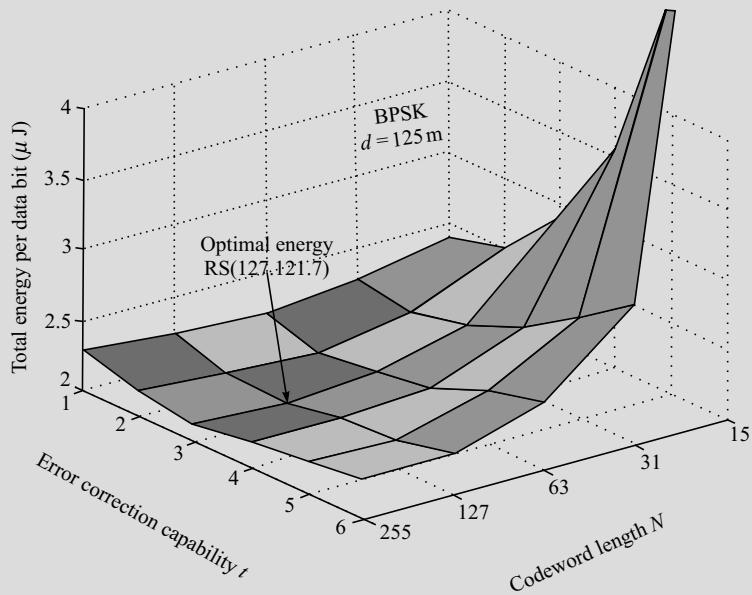


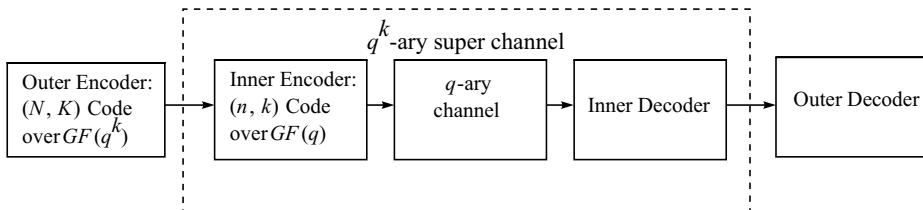
Fig. 5.6 Total energy versus the codeword length and the error correcting capability.

It is interesting to explore both the codeword length (block length), n and the error correcting capability t to achieve the minimum energy configuration. Figure 5.6 gives the simulation results for the StrongArm SA1100 processor, transmitter-receiver separation of 125 m and BPSK modulation. For this particular scenario, RS(127, 121, 7) results in the minimum energy.

5.10 Nested Codes

LO 3

One of the ways to achieve codes with large blocklengths is to *nest* codes. This technique combines a code of a small alphabet size and a code of a larger alphabet size. Let a block of q -ary symbols be of length kK . This block can be broken up into K sub-blocks of k symbols. Each sub-block can be viewed as an element of a q^k -ary alphabet. A sequence of K such sub-blocks can be encoded with an (N, K) code over $GF(q^k)$. Now, each of the $N q^k$ -ary symbols can be viewed as k q -ary symbols and can be coded with an (n, k) q -ary code. Thus, a nested code has two distinct levels of coding. This method of generating a nested code is given in Fig. 5.7.



Playing time: Maximum 74.7 min.

Disc specifications: Diameter 120 mm, thickness 1.2 mm, track pitch 1.6 μ m, one side medium, disc rotates clockwise, signal is recorded from inside to outside, constant linear velocity (CLV), recording maximises recording density (the speed of revolution of the disc is not constant; it gradually decreases from 500 to 200 r/min), pit is about 0.5 μ m wide, each pit edge is ‘1’ and all areas in between, whether inside or outside a pit, are ‘0’s.

Error correction: A typical error rate of a CD system is 10^{-5} , which means that a data error occurs roughly 20 times per second (bit rate \times BER). About 200 errors/s can be corrected.

Sources of errors: Dust, scratches, fingerprints, pit asymmetry, bubbles or defects in substrate, coating defects and dropouts.

Cross Interleave Reed-Solomon Code (CIRC)

- C2 can effectively correct burst errors.
 - C1 can correct random errors and detect burst errors.
 - Three interleaving stages to encode data before they are placed on a disc.
 - Parity checking to correct random errors.
 - Cross interleaving to permit parity to correct burst errors.
1. Input stage: 12 words (16-bit, 6 words per channel) of data per input frame divided into 24 symbols of 8 bits.
 2. C2 Reed-Solomon code: 24 symbols of data are enclosed into a (28, 24) RS code and 4 parity symbols are used for error correction.
 3. Cross interleaving: To guard against the burst errors, separate error correction codes, one code can check the accuracy of another, error correction is enhanced.
 4. C1 Reed-Solomon code: Cross-interleaved 28 symbols of the C2 code are encoded again into a (32, 28) R-S code (4 parity symbols are used for error correction).
 5. Output stage: Half of the code word is subject to a 1-symbol delay to avoid 2-symbol error at the boundary of symbols.

Performance of CIRC: Both RS coders (C1 and C2) have four parities, and their minimum distance is five. If error location is not known, up to two symbols can be corrected. If the errors exceed the correction limit, they are concealed by interpolation. Since even-numbered sampled data and odd-numbered sampled data are interleaved as much as possible, CIRC can conceal long burst errors by simple linear interpolation.

- Maximum correctable burst length is about 4000 data bits (2.5 mm track length).
- Maximum correctable burst length by interpolation in the worst case is about 12320 data bits (7.7 mm track length).

Sample interpolation rate is one sample every 10 hours at BER (Bit Error Rate) = 10^{-4} and 1000 samples at BER = 10^{-3} . Undetectable error samples (clicks) are less than one every 750 hours at BER = 10^{-3} and negligible at BER = 10^{-4} .

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 3:

1. Find the generator polynomial for the triple error correcting RS code of blocklength 15 over $GF(16)$. What is the code rate of this RS code? M
2. Find the generator polynomial for the triple error correcting RS code of blocklength 255 over $GF(2^8)$. What is the code rate of this RS code? M
3. Find the blocklength, n , of an RS code that has $k = t$. What is the code rate? M

If you have successfully solved the above problems,
you have mastered LO 3. Congratulations!



: The answers are available here. Scan to check



: Or

: Visit <http://qrcode.flipick.com/index.php/604>

5.11 Concluding Remarks

The class of BCH codes was discovered independently by Hocquenghem in 1959 and Bose and Ray Chaudhuri in 1960. The BCH codes constitute one of the most important and powerful classes of linear block codes, which are cyclic.

The Reed-Solomon codes were discovered by Irving S. Reed and Gustave Solomon who published a five-page paper in the *Journal of the Society for Industrial and Applied Mathematics* in 1960. They were staff members at MIT's Lincoln Laboratory at that time. The paper was titled "Polynomial Codes over Certain Finite Fields". Despite their advantages, Reed-Solomon codes did not go into use immediately after their invention. They had to wait for the hardware technology to catch up. In 1960, there was no such thing as fast digital electronics, at least not by today's standards. The Reed-Solomon paper suggested some nice ways to process data, but nobody knew if it was practical or not, and in 1960 it probably wasn't practical.

Eventually technology did catch up, and numerous researchers began to work on implementing the codes. One of the key individuals was Elwyn Berlekamp, a professor of electrical engineering at the University of California at Berkeley, who invented an efficient algorithm for decoding the Reed-Solomon code. Berlekamp's algorithm was used by Voyager II and is the basis for decoding in CD players. Many other bells and whistles (some of fundamental theoretic significance) have also been added. CDs, for example, use a version called cross-interleaved Reed-Solomon code, or CIRC. RS codes are also commonly used in wireless channels where burst errors are encountered.

LEARNING OUTCOMES

- A subfield of a field is a subset of the field, which is also a field. Thus, the subfield is ‘contained’ in the original field. Alternately, we can say that there is a base field, which is contained in the extension field. The base field is also called a ground field.
- A primitive element of $GF(q)$ is an element α such that every field element, except zero, can be expressed as a power of α . A field can have more than one primitive element.
- For an element $\beta \in GF(q)$, let n be the smallest integer such that $\beta^n = 1$, then n is the order of β . The order of a primitive element is $q - 1$.
- A primitive polynomial $p(x)$ over $GF(q)$ is a prime polynomial over $GF(q)$ with the property that in the extension field constructed modulo $p(x)$, the field element represented by x is a primitive element.
- A blocklength n of the form $n = q^m - 1$ is called a primitive block length for a code over $GF(q)$. A cyclic code over $GF(q)$ of primitive blocklength is called a primitive cyclic code.
- It is possible to factor $x^{q^m-1} - 1$ in the extension field $GF(q^m)$ to get $x^{q^m-1} - 1 = \prod_j (x - \beta_j)$, where β_j ranges over all the non-zero elements of $GF(q^m)$. This implies that each of the polynomials $f_i(x)$ can be represented in $GF(q^m)$ as a product of some of the linear terms, and each β_j is a zero of exactly one of the $f_i(x)$. This $f_i(x)$ is called the minimal polynomial of β_j .
- Suppose q is a prime, then an irreducible m^{th} degree polynomial, $f(x)$ with coefficients in $GF(q)$ divides $x^{q^m-1} - 1$.
- Two elements of $GF(q^m)$ that share the same minimal polynomial over $GF(q)$ are called conjugates with respect to $GF(q)$.
- BCH codes defined over $GF(q)$ with blocklength $q^m - 1$ are called primitive BCH codes.
- To determine the generator polynomial of a t -error correcting BCH code for a primitive blocklength $n = q^m - 1$, (i) choose a prime polynomial of degree m and construct $GF(q^m)$, (ii) find $f_i(x)$, the minimal polynomial of α^i for $i = 1, \dots, p$. (iii) obtain the generator polynomial $g(x) = \text{LCM} [f_1(x), f_2(x), \dots, f_{2t}(x)]$. Codes designed in this manner can correct at least t errors. In many cases the codes will be able to correct more than t errors. For this reason, $d = 2t + 1$ is called the designed distance of the code, and the minimum distance $d^* \geq 2t + 1$.
- Steps for decoding BCH codes:
 - (1) As a trial value, set $v = t$ and compute the determinant of the matrix of syndromes, \mathbf{M} . If the determinant is zero, set $v = t - 1$. Again compute the determinant of \mathbf{M} . Repeat this process until a value of v is found for which the determinant of the matrix of syndromes is non-zero. This value of v is the actual number of errors that occurred.
 - (2) Invert the matrix \mathbf{M} and find the coefficients of the error locator polynomial $\Lambda(x)$.
 - (3) Solve $\Lambda(x) = 0$ to obtain the zeroes and from them compute the error locations X_1, X_2, \dots, X_v . If it is a binary code, stop (because the magnitudes of error are unity).
 - (4) If the code is not binary, go back to the system of equations:

$$\begin{aligned}
 S_1 &= Y_1X_1 + Y_2X_2 + \dots + Y_vX_v \\
 S_2 &= Y_1X_1^2 + Y_2X_2^2 + \dots + Y_vX_v^2 \\
 &\vdots && \vdots \\
 S_{2t} &= Y_1X_1^{2t} + Y_2X_2^{2t} + \dots + Y_vX_v^{2t}
 \end{aligned}$$

Since the error locations are now known, these form a set of $2t$ linear equations. These can be solved to obtain the error magnitudes.

- The generator polynomial for a t -error correcting RS code will be simply $g(x) = \text{LCM}[f_1(x)f_2(x), \dots, f_{2t}(x)] = (x - \alpha)(x - \alpha^2) \dots (x - \alpha^{2t-1})(x - \alpha^{2t})$. Hence, the degree of the generator polynomial will always be $2t$. Thus, the RS code satisfies $n - k = 2t$.
- A Reed-Solomon code is a maximum distance separable (MDS) code and its minimum distance is $n - k + 1$.
- An (n, k) RS code can be punctured by deleting any one of its symbols to obtain a $(n - 1, k)$ code. An (n, k) RS code can be extended by adding parity check symbols. A single-extended RS code is also an MDS code.
- One of the ways to achieve codes with large blocklengths is to nest codes. This technique combines a code of a small alphabet size and a code of a larger alphabet size. Let a block of q -ary symbols be of length kK . This block can be broken up into K sub-blocks of k symbols. Each sub-block can be viewed as an element of a q^k -ary alphabet.

“Once you eliminate the impossible, whatever remains, no matter how improbable, must be the truth.”

Sherlock Holmes (by Sir Arthur Conan Doyle, 1859-1930)



MULTIPLE CHOICE QUESTIONS

5.1 Which of the following is true:

- (a) A primitive element of $GF(q)$ is an element α such that every field element except zero can be expressed as a power of α
- (b) A field can have more than one primitive element
- (c) Both (a) and (b)
- (d) None of the above

5.2 A primitive blocklength is

- (a) $n = q^m$
- (b) $n = q^m - 1$
- (c) $n = q^m + 1$
- (d) $n = q^{m-1} - 1$

5.3 Two elements of $GF(q^m)$ that share the same minimal polynomial over $GF(q)$ are called

- (a) Conjugates
- (b) Co-primes
- (c) Factors
- (d) None of the above

5.4 The generator polynomial for a t -error correcting BCH code is given by

- (a) $g(x) = f_1(x) \times f_2(x) \times \dots \times f_{2t}(x)$
- (b) $g(x) = \text{LCM}[f_1(x), f_2(x), \dots, f_{2t}(x)]$
- (c) $g(x) = f_1(x) + f_2(x) + \dots + f_{2t}(x)$
- (d) None of the above

5.5 The designed distance of a BCH code is

- (a) $d = 2t - 1$
- (b) $d = t + 1$
- (c) $d = t - 1$
- (d) $d = 2t + 1$

5.6 The generator polynomial for a t -error correcting RS code is

- (a) $g(x) = (x - 1)(x - \alpha) \dots (x - \alpha^{2t-1})(x - \alpha^{2t})$
- (b) $g(x) = (x - \alpha)(x - \alpha^2) \dots (x - \alpha^{2t-1})(x - \alpha^{2t})$
- (c) $g(x) = (x - \alpha)(x^2 - \alpha) \dots (x^{2t-1} - \alpha)(x^{2t} - \alpha)$
- (d) None of the above

5.7 RS code satisfies

- | | |
|------------------|-----------------------|
| (a) $n + k = 2t$ | (b) $n - k = t$ |
| (c) $n - k = 2t$ | (d) None of the above |

5.8 An RS code

- (a) is a maximum distance separable (MDS) code
- (b) has a minimum distance of $n - k + 1$
- (c) Both (a) and (b)
- (d) None of the above

5.9 In RS code, the blocklength is _____ less than number of symbols in symbol set (q)

- (a) One
- (b) Two
- (c) Three
- (d) $n - q$

5.10 Decoding of RS code comprises the determination of error

- | | |
|----------------------|-----------------------|
| (a) Location | (b) Magnitude |
| (c) Both (a) and (b) | (d) None of the above |

For interactive quiz with answers, scan the QR code given here

Or

Visit <http://qr-code.flipick.com/index.php/570>



SHORT-ANSWER TYPE QUESTIONS



- 5.1 Can the irreducible polynomial $x^2 + 1$ over real field be factored over a complex field?
- 5.2 If β_1 has order m and β_2 has order n , and $\gcd(m, n) = 1$ (i.e., relative primes), then the order of $\beta_1\beta_2$ will be mn . True or false?
- 5.3 Is the polynomial $x^3 + x^2 + 2$ irreducible over $GF(3)$?
- 5.4 Are $n = 342$ and $n = 279840$ primitive blocklengths?
- 5.5 Let α be the primitive element of $GF(2^6)$. Let $\beta = \alpha^7$. Find β^9 .
- 5.6 Does $(x^3 + x + 1)$ divide $(x^7 + 1)$ in $GF(2)$? Why/why not?
- 5.7 Find the square root of the element α^5 in $GF(2^3)$? That is, find γ where $\gamma^2 = \alpha^5$.
- 5.8 Can a binary $(9, 2)$ cyclic code have the generator polynomial $g(x) = x^7 + x^6 + x^4 + x^3 + x + 1$?
- 5.9 Consider Chen's cyclic code with $(n, k, d^*) = (63, 28, 15)$. Compare it with a BCH code of blocklength $n = 63$ and $t = 7$.
- 5.10 An RS $(255, 245)$ code can correct 6 errors. True or false?
- 5.11 For what value of n will an RS code have $k = t = 5$?
- 5.12 RS codes are cyclic codes. True or false?
- 5.13 What is the relationship between the degree of the generator polynomial of an RS code and the number of errors it can correct?

5.14 A single-extended RS code is also an MDS code. True or false?

5.15 Can an (n, k) RS code be punctured by deleting any one of its symbols to obtain a $(n - 1, k)$ code?

For answers, scan the QR code given here



Or

Visit <http://qrcode.flipick.com/index.php/605>

PROBLEMS



- M** 5.1 Construct $GF(9)$ from $GF(3)$ using an appropriate primitive polynomial.
- M** 5.2 (i) Find the generator polynomial $g(x)$ for a double error correcting *ternary* BCH code of blocklength 8. What is the code rate of this code? Compare it with the $(11, 6)$ ternary Golay code with respect to the code rate and the minimum distance.
(ii) Next, find the generator polynomial $g(x)$ for a triple error correcting *ternary* BCH code of blocklength 26.
- M** 5.3 Find the generator polynomial $g(x)$ for a binary single error correcting BCH code of blocklength 31. Use the primitive polynomial $p(x) = x^5 + x^2 + 1$ to construct $GF(32)$. What is the minimum distance of this code?
- S** 5.4 Determine the value of k for all binary BCH codes of blocklength 31.
- S** 5.5 Find the generator polynomial of the $(1023, k)$ binary BCH code with designed error-correcting capability $t = 4$.
- S** 5.6 Consider a double-error-correcting BCH code with $n = 31$ over $GF(2)$. Decode the received polynomials (a) $x^{28} + 1$ and (b) $x^{28} + x^{17} + 1$.
- S** 5.7 What are the possible generator polynomials for an RS code over $GF(2)$?
- S** 5.8 Find the generator polynomials and the minimum distance for the following codes:
 - (i) RS $(15, 11)$ code
 - (ii) RS $(15, 7)$ code
 - (iii) RS $(31, 21)$ code.
- M** 5.9 Consider the triple error correcting BCH $(15, 5)$ code over $GF(16)$. Given the received polynomial $r(x) = x^7 + x^5 + x^2$, what is the transmitted polynomial?
- M** 5.10 Show that every BCH code is a subfield subcode of a Reed-Solomon code of the same designed distance. Under what condition is the code rate of the BCH code equal to that of the RS code?
- D** 5.11 Show that the Vandermonde matrix, defined as

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ X_1 & X_2 & \cdots & X_\mu \\ X_1^2 & X_2^2 & \cdots & X_\mu^2 \\ \vdots & \vdots & \ddots & \vdots \\ X_1^{\mu-1} & X_2^{\mu-1} & \cdots & X_\mu^{\mu-1} \end{bmatrix} \quad (5.29)$$

has a non-zero determinant if and only if, all of the X_i for $i = 1, 2, \dots, \mu$ are distinct.

- M** 5.12 Consider the code over $GF(11)$ with a parity check matrix.

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 3 & \dots & 10 \\ 1 & 2^2 & 3^2 & \dots & 10^2 \\ 1 & 2^3 & 3^3 & \dots & 10^3 \\ \vdots & & & & \vdots \\ 1 & 2^8 & 3^8 & \dots & 10^8 \end{bmatrix} \quad (5.30)$$

- (i) Find the minimum distance of this code.
- (ii) Show that this is an optimal code with respect to the Singleton bound.

- M** 5.13 Consider the code over $GF(11)$ with a parity check matrix

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 3 & 4 & \dots & 10 \\ 1 & 2^2 & 3^2 & 4^2 & \dots & 10^2 \\ 1 & 2^3 & 3^3 & 4^3 & \dots & 10^3 \\ 1 & 2^4 & 3^4 & 4^4 & \dots & 10^4 \\ 1 & 2^5 & 3^5 & 4^5 & \dots & 10^5 \end{bmatrix} \quad (5.31)$$

- (i) Show that the code is a triple error correcting code.
- (ii) Find the generator polynomial for this code.

- S** 5.14 The ECMA standard for high speed Ultra Wideband (UWB) communication uses the RS(255, 249) code over $GF(2^8)$.
- (i) Find the generator polynomial for this code.
 - (ii) Give the shift-register based implementation of the systematic RS(255, 249) code.
- M** 5.15 Give the generator polynomials for a double- and triple-error correcting RS code over $GF(2^5)$. For the triple-error correcting RS code, given the received polynomial $r(x) = \alpha^5x^{18} + \alpha^{11}x^{15} + \alpha^9$, what is the transmitted polynomial?
- M** 5.16 For deep-space communication, we wish to transmit color pictures with 2048 colors using an RS code. How many errors in a codeword can be corrected if we design the code over
 - (i) $GF(16)$
 - (ii) $GF(32)$?
- D** 5.17 Consider the RS code of blocklength 7 over $GF(2^3)$ with the generator polynomial $g(x) = x^4 + \alpha^3x^3 + x^2 + \alpha x + \alpha^3$.
- (i) Find the parity check polynomial, $h(x)$.
 - (ii) How many errors can it correct?
 - (iii) Decode the received word $r(x) = \alpha^6x^6 + \alpha x^5 + \alpha^4x^4 + \alpha^3x^3 + \alpha^4x^2 + \alpha x + 1$.
- D** 5.18 Consider the RS (255, 251) code over $GF(2^8)$ constructed using primitive polynomial $p(x) = x^8 + x^4 + x^3 + x^2 + 1$. Find the transmitted codeword given the syndromes $S_1 = \alpha^{81}$, $S_2 = \alpha^{55}$, $S_3 = \alpha^{234}$, $S_4 = \alpha^{163}$.

- D** 5.19 Show that the dual of an RS code is also an RS code. What are the parameters of this dual code?
- M** 5.20 Show that the $(2^m - 1, k, d^*)$ RS code contains the binary BCH code of blocklength $n = 2^m - 1$. What is the minimum distance of this BCH code?

COMPUTER PROBLEMS



- 5.1 Write a computer program which takes in the coefficients of a primitive polynomial, the values of q and m , and then constructs the extension field $GF(q^m)$.
- 5.2 Write a computer program that performs addition and multiplication over $GF(2^m)$, where m is an integer.
- 5.3 Write a program that takes in a polynomial with coefficients in $GF(q^m)$ and returns whether it is irreducible?
- 5.4 Find the generator polynomial $g(x)$ for a binary BCH code of blocklength 63. Use the primitive polynomial $p(x) = x^6 + x + 1$ to construct $GF(64)$. What is the minimum distance of this code?
- 5.5 Write a program that performs BCH decoding given n, q, t and the received vector.
- 5.6 Write a program that outputs the generator polynomial of the Reed-Solomon code with the codeword length n and the message length k . A valid n should be $2^M - 1$, where M is an integer not less than 3. The program should also list the minimum distance of the code.
- 5.7 Write a computer program that performs the two level RS coding as done in a standard CD.
- 5.8 Write a computer program for the systematic encoder for RS(63, 55) over $GF(2^6)$ using the shift-register implementation.

PROJECT IDEAS

- 5.1 **Hybrid RS-ARQ Scheme:** Design an RS scheme in conjunction with Automatic Repeat request (ARQ) protocol to improve transmission reliability in wireless networks. Assume that it is a packet-based communication and packet erasure rate is 10% in the wireless channel. What would be the parameters of the RS scheme?
- 5.2 **Fountain Codes:** Fountain codes are useful for transmitting information over computer networks. As soon as a receiver requests data, the packets are copied and forwarded to the recipient. The copying can be done actively by the sender, or it can be done by the network. The recipient collects the output symbols, and stops as soon as it has received n of them. Design an encoder and decoder for Fountain codes. Compare its performance with that of RS codes.
- 5.3 **BCH Codes in ITU-T Rec. H.261:** BCH (511, 493) is used in the ITU telecommunication standard H.261(for video compression). The generator polynomial is given by $p(x) = x^{18} + x^{15} + x^{12} + x^{10} + x^8 + x^7 + x^6 + x^3 + 1$. Give the circuit implementation of this BCH (511, 493) code. What is the significance of $n = 511$ from video coding perspective? How many errors can it correct? Test the efficacy of this BCH (511, 493) code when video is transmitted over a BSC with $p = 10^{-2}$.

REFERENCES FOR FURTHER READING

1. Roth, R.M., *Introduction to Coding Theory*, Cambridge University Press, 2006.
2. Lin, S. and Costello, D.J., *Error Control Coding: Fundamentals and Applications* (2nd edition), Prentice-Hall, 2004.
3. Blahut, R.E., *Algebraic Codes for Data Transmission*, Cambridge University Press, 2002.
4. Moon, T.K., *Error Correction Coding: Mathematical Methods and Algorithms*, Wiley, 2005.
5. Morelos-Zaragoza, R.H., *The Art of Error Correcting Coding*, Wiley & Sons, 2006.

Space-Time Codes

It is dangerous to put limit on wireless.

Guglielmo Marconi (1874-1937)

Learning objectives

After studying this chapter the students will be able to:

- LO 1** Know the basic idea behind space-time block codes, diversity gain, coding gain, the rank and determinant criteria for designing space-time block codes.
- LO 2** Define real orthogonal design, generalised real orthogonal design, complex orthogonal design, generalised complex orthogonal design and quasi-orthogonal space-time block codes.
- LO 3** Explain the space-time block code design targets and carry out performance analysis.

6.1 Introduction to Space-Time Codes

An inherent characteristic of a wireless channel is multipath propagation, which leads to signal fading. One of the ways to improve the capacity of wireless channels is to use multiple antennas at the transmitter and receiver. However, in many real-world systems, the receiver is required to be small by design (cell-phones, personal digital assistants, smart-phones, etc.). In such physically small receivers, it is impractical to put multiple antennas. Recall that the different elements, in a multi-antenna system, need a minimum separation to work well. Therefore, it is not practical to deploy multiple antennas at the receiver. On the other hand, the base station can easily accommodate multiple antennas, thus making multiple transmit antennas possible. Hence, we need to explore Multi Input Single Output (MISO) systems that can provide diversity. Space-time codes directly address this requirement.

...
This chapter comes with a video overview by the author. Scan here to know more or
Visit <http://qrcode.flipick.com/index.php/611>



In Chapter 3, we have given a brief introduction to Space Time Block Codes (STBC). In this chapter we will study the wonderful properties of this new class of codes. As discussed earlier, a space time code is a mapping from input bits to the symbols being transmitted. However, we first need to understand what a “good” code is. This will provide us with some guidelines for designing good space-time codes.

In this chapter

We will start by defining the anatomy of a STBC. We will take a specific example of the Alamouti code to explain the concept. Then the pairwise error probability, diversity gain and coding gain will be introduced. Subsequently, the rank and determinant criteria for designing space-time block codes will be discussed, in LO 1.

Next, we will study real orthogonal design, generalised real orthogonal design, complex orthogonal design, generalised complex orthogonal design and quasi-orthogonal design for space-time block codes, in LO 2.

Finally, we will conclude with a brief discussion on the design targets for space-time block codes and their performance, in LO 3.

6.2 Anatomy of Space-Time Block Code

LO 1



Know the basic idea behind space-time block codes, diversity gain, coding gain, the rank and determinant criteria for designing space-time block codes.

The basic structure of a space-time block encoder is shown in Fig. 6.1. The input of the first block is a sequence of b bits and the output of the block is a sequence of K symbols (complex or real). This sequence of symbols is then spread in space (over the N transmit antennas) and in time (over T time slots) in order to form a codeword represented by a matrix \mathbf{C} of size $N \times T$. We can normalise the codeword average transmit power such that $E[\text{Tr}(\mathbf{CC}^H)] = T$, where $\text{Tr}(\cdot)$ is the trace and $E[\cdot]$ is the expectation. The basic idea is to spread information symbols in space and time in order to improve either the diversity gain, or the spatial multiplexing rate, or both the diversity gain and the spatial multiplexing rate. We assume that the channel is constant over the duration of the STBC codeword T . It should be kept in mind that STBCs are usually concatenated with outer coding in order to improve error correcting capability of the system.

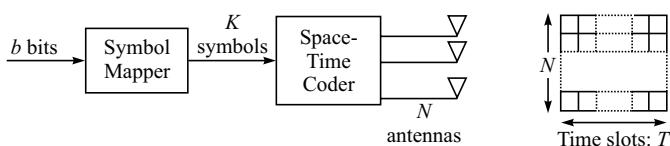


Fig. 6.1 Illustration of a general space-time encoder.

In order to understand the basic idea behind a space-time block code, we first look at a simple example.

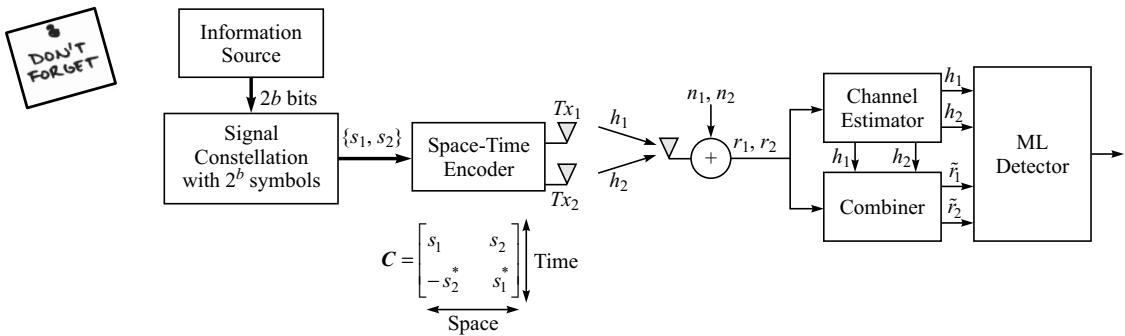


Fig. 6.2 Illustration of a simple space-time block code.

Example 6.1 Consider a wireless system consisting of two transmit antennas and one receive antenna, as shown in Fig. 6.2. The information source generates bits, which are mapped to symbols from a signal constellation (MPSK, MQAM, etc.). The signal constellation can be real or complex constellation. To transmit b bits/cycle, we use a modulation scheme that maps one symbol from a constellation with 2^b symbols. The output of the mapper, $\{x_1, x_2\}$, is input to a space-time coding block. The encoding takes in two time slots (the time axis) over two antenna elements (the space axis). In each encoding operation, the encoder takes two symbols s_1 and s_2 and transmits according to the following *coding strategy*, depicted by the matrix

$$\mathbf{X} = \begin{bmatrix} x_1 & x_2 \\ -x_2^* & x_1^* \end{bmatrix} \quad (6.1)$$

where x_1^* is the complex conjugate of x_1 . For example, if symbols s_1 and s_2 are selected, we map $x_1 \rightarrow s_1$ and $x_2 \rightarrow s_2$ and obtain the codeword matrix as

$$\mathbf{C} = \begin{bmatrix} s_1 & s_2 \\ -s_2^* & s_1^* \end{bmatrix} \quad (6.2)$$

A frame of data lasts two symbol periods. During symbol period 1, antenna 1 and antenna 2 transmit s_1 and s_2 respectively. During symbol period 2, antenna 1 and antenna 2 transmit $-s_2^*$ and s_1^* respectively.

For better visualisation, we can write

	Antenna 1	Antenna 2
Time period 1	s_1	s_2
Time period 2	$-s_2^*$	s_1^*

From (6.1) we note that

$$\mathbf{X}^H \mathbf{X} = (|x_1|^2 + |x_2|^2) \mathbf{I}_2 \quad (6.3)$$

where \mathbf{I}_2 is a 2×2 identity matrix.

The channel from antenna 1 to the receiver (h_1) and antenna 2 to the receiver (h_2) can be modelled as

$$\begin{aligned} h_1 &= \alpha_1 e^{-j\phi_1} \\ h_2 &= \alpha_2 e^{-j\phi_2} \end{aligned} \quad (6.4)$$

It is assumed that the fading channel coefficients h_1 and h_2 are constant across two consecutive symbol periods (is that a fair assumption?). The received signal over the first symbol period, denoted by r_1 , can be expressed as

$$r_1 = h_1 s_1 + h_2 s_2 + n_1 \quad (6.5)$$

Similarly, the signal over the second symbol period, denoted by r_2 , can be expressed as

$$r_2 = -h_1 s_2^* + h_2 s_1^* + n_2 \quad (6.6)$$

where n_1 and n_2 are AWGN samples for the two symbol periods respectively.

The aim of the receiver is to extract the transmitted symbols (s_1 and s_2) from r_1 and r_2 . The receiver combines the received signal according to the following *combining scheme*

$$\begin{aligned} \tilde{r}_1 &= h_1^* r_1 + h_2^* r_2 \\ \tilde{r}_2 &= h_2^* r_1 - h_1^* r_2 \end{aligned} \quad (6.7)$$

Note that the receiver must have an estimate of h_1 and h_2 in order to implement the combining scheme. As explicitly stated in (6.4), both h_1 and h_2 are complex. Substituting (6.5) and (6.6) in (6.7) we obtain

$$\begin{aligned} \tilde{r}_1 &= (|h_1|^2 + |h_2|^2) s_1 + h_1^* n_1 + h_2^* n_2 \\ \tilde{r}_2 &= (|h_1|^2 + |h_2|^2) s_2 - h_1^* n_2 + h_2^* n_1 \end{aligned} \quad (6.8)$$



The beauty of (6.8) is that \tilde{r}_1 depends only on s_1 , and not on s_2 . So, the detection can be carried out with respect to this single quantity! Similarly, \tilde{r}_2 depends only on s_2 , and not on s_1 . The detector uses the maximum likelihood decision rule on each of the signals separately

$$\begin{aligned} \tilde{s}_1 &= \arg \min_{s \in S} |\tilde{r}_1 - (|h_1|^2 + |h_2|^2)s| \\ \tilde{s}_2 &= \arg \min_{s \in S} |\tilde{r}_2 - (|h_1|^2 + |h_2|^2)s| \end{aligned} \quad (6.9)$$

The example of the space-time block code described in Example 6.1 is called the **Alamouti code**. The scheme sends two symbols over two time periods, thereby making the rate of the code equal to 1. At the same time, it also provides a diversity of 2.

Next, consider the following example.

Example 6.2 A wireless system consisting of two transmit antennas and one receive antenna, as shown in Fig. 6.2. Let us employ a different combining scheme that leads to values that are a mixture of the transmitted signal. For example, instead of (6.8), we obtain:

$$\begin{aligned} \tilde{r}_1 &= as_1 + bs_2 + \tilde{n}_1 \\ \tilde{r}_2 &= cs_1 + ds_2 + \tilde{n}_2 \end{aligned} \quad (6.10)$$

where a, b, c and d are some coefficients. In this case,

$$\tilde{\mathbf{r}} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \begin{bmatrix} \tilde{n}_1 \\ \tilde{n}_2 \end{bmatrix} \quad (6.11)$$

Let us say,

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad (6.12)$$

Then, the ML decision rule can be written as

$$[\tilde{s}_1 \quad \tilde{s}_2] = \arg \min_{s \in S^2} \|\tilde{\mathbf{r}} - \mathbf{As}\|^2 \quad (6.13)$$



This process of minimisation is basically a search for a vector of length 2. If the constellation has M points, then the computational complexity for the search is $O(M^2)$. If, the number of transmit antenna is increased to N , the computational complexity for the search will be $O(M^N)$. For the Alamouti code, this increase in the complexity is avoided because of the use of an orthogonal encoding matrix, as depicted by (6.3).

Example 6.3 Consider a 2×1 wireless system that employs the Alamouti code with QPSK modulation.

Let the input bit-stream be 1 1 0 0 0 0 1 0 1 0 0 1

Assuming the constellation has 2^b symbols, $b = 2$. In Alamouti code, $K = 2$ symbols are transmitted over 2 time periods. So we take $2 \times 2 = 4$ bits at a time and choose two corresponding symbols.

Let us assume Gray coding: $00 \rightarrow s_0$, $01 \rightarrow s_1$, $11 \rightarrow s_2$, $10 \rightarrow s_3$.

Pertaining to the first 4 bits [11 00] we pick the two symbols $[s_2 \ s_0]$.

We now use Alamouti code to transmit the symbols in the following manner.

	Antenna 1	Antenna 2
Time period 1	s_2	s_0
Time period 2	$-s_0^*$	s_2^*

For the subsequent 4 bits [00 10] we have the two symbols $[s_0 \ s_3]$. Again, we use Alamouti code to get

	Antenna 1	Antenna 2
Time period 3	s_0	s_3
Time period 4	$-s_3^*$	s_0^*

For the last 4 bits: [10 01] we have the two symbols $[s_3 \ s_1]$. Again, we use Alamouti code to get

	Antenna 1	Antenna 2
Time period 5	s_3	s_1
Time period 6	$-s_1^*$	s_3^*

In the previous examples, we have used 2 transmit antennas and two time periods, leading to a 2×2 code matrix, as given by (6.2). In general, consider a wireless system that uses N transmit antennas during T time periods. Then the space-time code can be represented by an $N \times T$ matrix



$$\mathbf{C} = \begin{bmatrix} C_{1,1} & C_{1,2} & \cdots & C_{1,N} \\ C_{2,1} & C_{2,2} & \cdots & C_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ C_{T,1} & C_{T,2} & \cdots & C_{T,N} \end{bmatrix} \quad (6.14)$$

6.3 Space-Time Code Design Criteria

LO 1

In Example 6.1 we saw that a space-time block code is essentially a mapping from input bits to the transmitted symbols. These symbols are transmitted simultaneously from the different antennas. The aim of the space-time decoder is to correctly guess the transmitted symbols. An error occurs when one codeword is wrongly mistaken as another codeword. Suppose a wireless system uses N transmit antennas during T time periods. Let the transmitted codeword be

$$\mathbf{C}^1 = \begin{bmatrix} C_{1,1}^1 & C_{1,2}^1 & \cdots & C_{1,N}^1 \\ C_{2,1}^1 & C_{2,2}^1 & \cdots & C_{2,N}^1 \\ \vdots & \vdots & \ddots & \vdots \\ C_{T,1}^1 & C_{T,2}^1 & \cdots & C_{T,N}^1 \end{bmatrix} \quad (6.15)$$

The decoder is said to make an error if it decides that a different codeword was indeed transmitted, say for example,

$$\mathbf{C}^2 = \begin{bmatrix} C_{1,1}^2 & C_{1,2}^2 & \cdots & C_{1,N}^2 \\ C_{2,1}^2 & C_{2,2}^2 & \cdots & C_{2,N}^2 \\ \vdots & \vdots & \ddots & \vdots \\ C_{T,1}^2 & C_{T,2}^2 & \cdots & C_{T,N}^2 \end{bmatrix} \quad (6.16)$$

Definition 6.1 The probability of erroneously decoding codeword \mathbf{C}^2 when codeword \mathbf{C}^1 was transmitted is called the **pairwise error probability**, and is denoted by $P(\mathbf{C}^1 \rightarrow \mathbf{C}^2)$.

Suppose, the codebook contains K codewords. We can use the union bound to find the upperbound on the probability of error that the codeword \mathbf{C}^1 was transmitted and erroneously decoded. The upperbound is given by

$$P(\text{error} | \mathbf{C}^1) \leq \sum_{i=2}^K P(\mathbf{C}^1 \rightarrow \mathbf{C}^i) \quad (6.17)$$

and will be used to determine the code design criteria. To calculate the pairwise error probability, we assume a fixed, known channel matrix, \mathbf{H} . The average error is calculated by averaging over the distribution of \mathbf{H} . After some algebra, the bound can be written as

$$P(\mathbf{C}^1 \rightarrow \mathbf{C}^2) \leq \left(\frac{1}{\prod_{i=1}^N (1 + \lambda_i E_s / 4N_0)} \right)^M \quad (6.18)$$

where, E_s/N_0 is the SNR, M is the number of receive antennas and λ_n , $n = 1, 2, \dots, N$ are the eigenvalues of the matrix $\mathbf{A}(\mathbf{C}^1, \mathbf{C}^2)$, defined as

$$\begin{aligned} \mathbf{A}(\mathbf{C}^1, \mathbf{C}^2) &= (\mathbf{C}^2 - \mathbf{C}^1)^H (\mathbf{C}^2 - \mathbf{C}^1) \\ &= D(\mathbf{C}^2, \mathbf{C}^1)^H D(\mathbf{C}^2, \mathbf{C}^1) \end{aligned} \quad (6.19)$$

where $D(\mathbf{C}^2, \mathbf{C}^1)$ is the difference matrix (error matrix). Let r be the rank of the matrix $\mathbf{A}(\mathbf{C}^1, \mathbf{C}^2)$. At high SNR, the one in the denominator of (6.18) can be neglected in order to obtain the following upperbound

$$P(\mathbf{C}^1 \rightarrow \mathbf{C}^2) \leq \frac{1}{\left(\prod_{i=1}^r \lambda_i \right)^M (E_s / 4N_0)^{rM}} \quad (6.20)$$

As discussed earlier in Chapter 3, the error probability can be expressed as

$$P(\mathbf{C}^1 \rightarrow \mathbf{C}^2) \approx \frac{c}{(G_c S)^{G_d}} \quad (6.21)$$

where S is the SNR, G_d is the diversity gain (or diversity order) and G_c is the coding gain and c is some constant. Comparing (6.20) and (6.21) we draw the following interesting conclusions

- The diversity gain of the code is $G_d = rM$, i.e., the product of the rank of the matrix $\mathbf{A}(\mathbf{C}^1, \mathbf{C}^2)$ and the number of receive antennas.
- The coding gain is a function of the product of the non-zero eigen values of the matrix $\mathbf{A}(\mathbf{C}^1, \mathbf{C}^2)$, or equivalently, the determinant of the matrix $\mathbf{A}(\mathbf{C}^1, \mathbf{C}^2)$.
- Full diversity (MN) is possible if the matrix $\mathbf{A}(\mathbf{C}^1, \mathbf{C}^2)$ is of full rank.

Definition 6.2 The **Coding Gain Distance** (CGD) between codewords is the product of the non-zero eigenvalues of the matrix $\mathbf{A}(\mathbf{C}^1, \mathbf{C}^2)$.

Definition 6.3 The **Rank and Determinant Criteria** for designing space-time block codes is given by

- 
- In order to achieve maximum diversity, the matrix $\mathbf{A}(\mathbf{C}^i, \mathbf{C}^j)$ should be of full rank for any two codewords, $\mathbf{C}^i \neq \mathbf{C}^j$. The smallest value of r , over any pair of codewords, provides a diversity gain of rM . [rank criteria]
 - In order to achieve maximum coding gain, the minimum determinant of the matrix $\mathbf{A}(\mathbf{C}^i, \mathbf{C}^j)$ should be maximised for any two codewords, $\mathbf{C}^i \neq \mathbf{C}^j$. [determinant criteria]

Definition 6.4 The **Spatial Multiplexing Rate**, or simply, the **Rate** of a space-time block code is defined

as $R = \frac{K}{T}$, where K distinct symbols are transmitted over T time intervals.

Let us revisit the Alamouti code discussed in Example 6.1. It is a coding scheme with $N = 2$ transmit antennas and the transmitted codeword is given by (6.2). Consider a different pair of symbols (s'_1, s'_2) with the corresponding codeword matrix

$$\mathbf{C}' = \begin{bmatrix} s'_1 & s'_2 \\ -s'^*_2 & s'^*_1 \end{bmatrix} \quad (6.22)$$

The difference matrix can be written as

$$D(\mathbf{C}, \mathbf{C}') = \begin{bmatrix} s' - s_1 & s'_2 - s_2 \\ s^*_2 - s'^*_2 & s'^*_1 - s^*_1 \end{bmatrix} \quad (6.23)$$

The determinant $\det[D(\mathbf{C}^2, \mathbf{C}^1)] = |s_1 - s'_1|^2 + |s_2 - s'_2|^2$ is zero if and only if $s_1 = s'_1$ and $s_2 = s'_2$. Thus, $D(\mathbf{C}^2, \mathbf{C}^1)$ is full rank when $\mathbf{C}^2 \neq \mathbf{C}^1$. Therefore, Alamouti code satisfies the rank criterion and gives a diversity of $2M$, where M is the number of receive antennas.

We can summarise the following about Alamouti code:



- Maximum Diversity: Since the code satisfies the rank criterion, it provides the maximum diversity (diversity gain = 2).
- Simple decoding: Each symbol can be decoded separately, using only linear processing.
- Full Rate: Alamouti code transmits two symbols in two time periods, thereby providing a rate of $R = 1$.

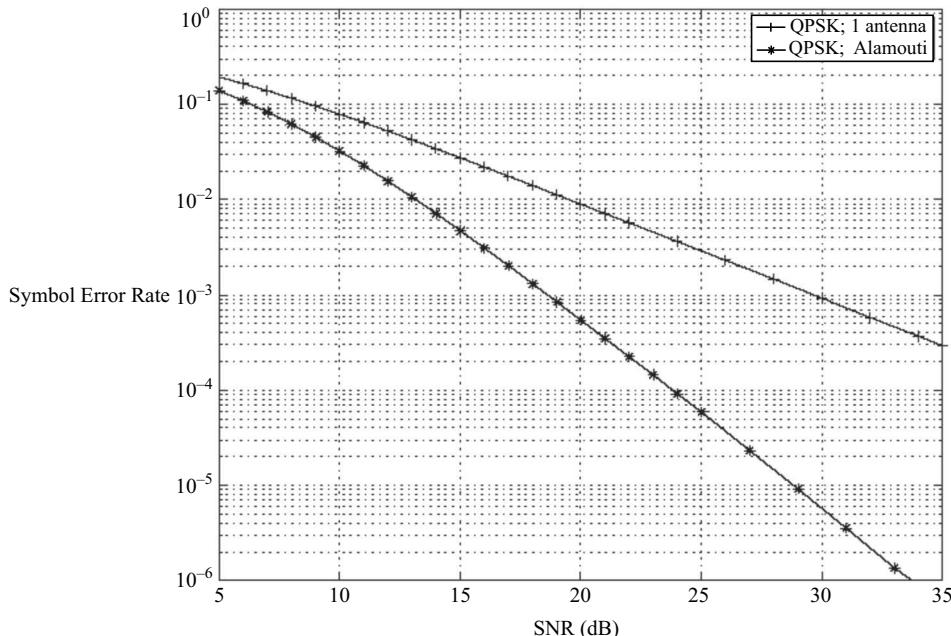


Fig. 6.3 The performance of Alamouti code, over a quasi-static, flat Rayleigh fading channel, using a QPSK constellation.



The performance of Alamouti code, over a quasi-static flat Rayleigh fading channel, is given in Fig. 6.3. The term ‘quasi-static’ implies that for two time periods the channel does not change. The figure gives the symbol error probability (SEP) versus SNR for $N = 2$ transmit antennas and $M = 1$ receive antenna. For comparison, the SEP for a system with only for $N = 1$ transmit antenna 1 and $M = 1$ receive antenna is also plotted. Both systems use the QPSK constellation. It is clear from the figure that the Alamouti code provides a much better performance due to the diversity gain (observe the slopes of the curves). From (6.20), the SEP decreases as S^{-2} , where S is the SNR. Thus, the effect of the diversity gain becomes more pronounced at higher SNRs. This is evident from the fact that the performance gap widens as the SNR increases (SEP decreases). For example, the performance gain for the Alamouti scheme is 11 dB at $\text{SEP} = 10^{-3}$ while the gap widens to 14 dB at $\text{SEP} = 2 \times 10^{-4}$.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 1:

1. Suppose we have a 2×1 wireless system that employs Alamouti code and uses QPSK modulation and the input bit-stream being $1\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ \dots$. What will be the transmitted symbols from each antenna?

2. Consider a 2×1 baseband communication system that uses the code $\mathbf{G}_2 = \begin{bmatrix} x_1 & x_2 \\ -x_2 & x_1 \end{bmatrix}$ in conjunction with 4-PAM represented by $\{-3V, -1V, 1V, 3V\}$. Let the input bit-stream be $1\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ \dots$ Sketch the transmitted signal from each transmitter.

If you have successfully solved the above problems,
you have mastered LO 1. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/607>



Levels of Difficulty

- S** Simple: Level 1 and Level 2 Category
- M** Medium: Level 3 and Level 4 Category
- D** Difficult: Level 5 and Level 6 Category

6.4 Real Orthogonal Design

LO 2



Define real orthogonal design, generalised real orthogonal design, complex orthogonal design, generalised complex orthogonal design and quasi-orthogonal space-time block codes.

We have seen in Example 6.1 that the optimal decisions can be made based on single symbol at a time, as given by (6.9). So, what is so unique about this code? Let us reconsider (6.5) and (6.6). The pair may be succinctly written as

$$(r_1 \quad r_2^*) = (s_1 \quad s_2)\Omega + (n_1 \quad n_2^*) \quad (6.24)$$

where

$$\Omega = \Omega(h_1 \quad h_2) = \begin{bmatrix} h_1 & h_2^* \\ h_2 & -h_1^* \end{bmatrix} \quad (6.25)$$

Note that complex conjugate of both sides of (6.6) has been taken prior to rewriting in the matrix form. Upon multiplying both sides of (6.24) by Ω^H , we obtain

$$(\tilde{s}_1 \quad \tilde{s}_2) = (r_1 \quad r_2^*)\Omega^H = (|h_1|^2 + |h_2|^2)(s_1 \quad s_2) + N \quad (6.26)$$

where N can also be represented as Gaussian noise. Equation (6.26) actually represents two separate equations that can be used for decoding the two transmitted signals using simple ML decoding.

Suppose, instead of two transmit antennas, there was a single transmit antenna, then the received signal would depend solely on h , the fading channel coefficient. In deep fading environment, $|h|^2$ would be very small, and the received signal would be dominated by the noise in the system. However, if we consider the system with two transmit antennas, the received signal would depend on $(|h_1|^2 + |h_2|^2)$, as shown by (6.26). For the system to be dominated by noise, the quantity $(|h_1|^2 + |h_2|^2)$ should be very small, which in turn implies that *both* $|h_1|^2$ and $|h_2|^2$ must be very small *simultaneously*. This suggests that both h_1 and h_2 must be fading at the same time. However, this is highly unlikely, as we have assumed h_1 and h_2 to be independent. Thus, the received signal is less likely to be in fade, because of the diversity provided by the two independently fading channels. Here we have argued, based on intuition, why the system with two transmit antennas provides diversity.

Further, observe that

$$\Omega\Omega^H = (|h_1|^2 + |h_2|^2)\mathbf{I}_2 \quad (6.27)$$

where \mathbf{I}_2 is a 2×2 identity matrix. Alternately, we can represent the **generator matrix** of the code by comparing with (6.1).

$$\mathbf{G} = \begin{bmatrix} x_1 & x_2 \\ -x_2^* & -x_1^* \end{bmatrix} \quad (6.28)$$

It is easy to see that

$$\mathbf{G}\mathbf{G}^H = (|x_1|^2 + |x_2|^2)\mathbf{I}_2 \quad (6.29)$$



As noted before, the beauty of this generator matrix is that it can decompose the decision problem at the receiver such that decisions can be made based on a single symbol at a time. This is possible using orthogonal design, defined as follows.

Definition 6.5 A Real Orthogonal Design of size N is an $N \times N$ matrix $\mathbf{G} = \mathbf{G}(x_1, x_2, \dots, x_N)$, with entries consisting of real elements drawn from $\pm x_1, \pm x_2, \dots, \pm x_N$ such that

$$\mathbf{G}^T \mathbf{G} = \left(\sum_{i=1}^N x_i^2 \right) \mathbf{I}_N \quad (6.30)$$

where \mathbf{I}_N is an $N \times N$ identity matrix. A real orthogonal design exists if and only if $N = 2, 4$ and 8 . We note that \mathbf{G} is proportional to an orthogonal matrix.

Example 6.4

Real orthogonal design for $N = 2$ is



$$\mathbf{G}_2 = \begin{bmatrix} x_1 & x_2 \\ -x_2 & x_1 \end{bmatrix} \quad (6.31)$$

for $N = 4$ is

$$\mathbf{G}_4 = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ -x_2 & x_1 & -x_4 & x_3 \\ -x_3 & x_4 & x_1 & -x_2 \\ -x_4 & -x_3 & x_2 & x_1 \end{bmatrix} \quad (6.32)$$

and for $N = 8$ is

$$\mathbf{G}_8 = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ -x_2 & x_1 & x_4 & -x_3 & x_6 & -x_5 & -x_8 & x_7 \\ -x_3 & -x_4 & x_1 & x_2 & x_7 & x_8 & -x_5 & -x_6 \\ -x_4 & x_3 & -x_2 & x_1 & x_8 & -x_7 & x_6 & -x_5 \\ -x_5 & -x_6 & -x_7 & -x_8 & x_1 & x_2 & x_3 & x_4 \\ -x_6 & x_5 & -x_8 & x_7 & -x_2 & x_1 & -x_4 & x_3 \\ -x_7 & x_8 & x_5 & -x_6 & -x_3 & x_4 & x_1 & -x_2 \\ -x_8 & -x_7 & x_6 & x_5 & -x_4 & -x_3 & x_2 & x_1 \end{bmatrix} \quad (6.33)$$

6.5 Generalised Real Orthogonal Design

LO 2

It is known that the real orthogonal designs exist only for $N = 2, 4$ and 8 . In order to have more designs, we generalise the orthogonal designs to non-square real matrices of size $T \times N$. In these designs, the number of time periods (T) is not necessarily equal to the number of transmit antenna elements (N).

Definition 6.6 A Generalised Real Orthogonal Design is a $T \times N$ matrix with entries consisting of real elements drawn from $\pm x_1, \pm x_2, \dots, \pm x_K$ such that

$$\mathbf{G}^T \mathbf{G} = \left(\sum_{i=1}^K x_i^2 \right) \mathbf{I}_N \quad (6.34)$$

where \mathbf{I}_N is an $N \times N$ identity matrix and K is a constant that represents the number of distinct symbols being transmitted.

The rate of \mathbf{G} is given as

$$R = \frac{K}{T} \quad (6.35)$$

Definition 6.7 A generalised real orthogonal design with rate $R = 1$ is called a **Full Rate Orthogonal Design**.

Since STBC from orthogonal designs have $T = K = N$, the rate $R = 1$, and they form a special case of the generalised real orthogonal design. Generalised real orthogonal designs provide full diversity and separate decoding of symbols.

A real space time block code is defined as one that uses \mathbf{G} as the **transmission matrix**. Let us assume that the transmission is being carried out using a constellation consisting of 2^b symbols. The steps involved in encoding using a generalised real orthogonal design are as follows:

- 1. Pick up a block of Kb bits from the input bit-stream.
- 2. Based on these Kb bits, select the K symbols from the constellation: (s_1, s_2, \dots, s_k)
- 3. Substitute $x_k \rightarrow s_k$ in the matrix \mathbf{G} and generate the codeword $\mathbf{C} = \mathbf{G}(s_1, s_2, \dots, s_k)$. Note that \mathbf{C} is a matrix of size $T \times N$. The transmission is done row-wise. Each row (of length N) is transmitted in one time period using N antenna elements simultaneously.
- 4. At time period $t = 1, 2, \dots, T$, transmit the t^{th} row of \mathbf{C} using the different antenna elements, $n = 1, 2, \dots, N$. Thus, the entry $C_{t,n}$ is transmitted from antenna element n at time period t .
- 5. At the end of T time periods, effectively K symbols would have been transmitted, thus justifying the definition $R = K/T$. Intuitively, T corresponds to the block length of the code.

The block diagram of the encoder is shown in Fig. 6.4.

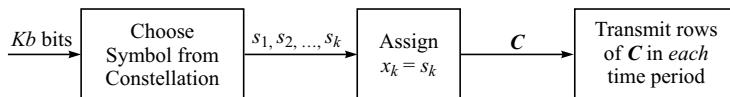


Fig. 6.4 Block diagram of the encoder for orthogonal space-time block code.

In order to maximise the rate, it is important to have the smallest value of the blocklength, T . This parameter, T , determines the decoding delay of the code, because we cannot start the decoding process until all the codewords have been received. This motivates us to define the following.

Definition 6.8 A generalised orthogonal design with minimum possible value of the blocklength T is called **Delay Optimal**.

Example 6.5 Consider the following 4×3 matrix

$$\mathbf{G} = \begin{bmatrix} x_1 & x_2 & x_3 \\ -x_2 & x_1 & -x_4 \\ -x_3 & x_4 & x_1 \\ -x_4 & -x_3 & x_2 \end{bmatrix} \quad (6.36)$$

Computing $\mathbf{G}^T \mathbf{G}$ gives

$$\mathbf{G}^T \mathbf{G} = \begin{bmatrix} x_1^2 + x_2^2 + x_3^2 + x_4^2 & 0 & 0 \\ 0 & x_1^2 + x_2^2 + x_3^2 + x_4^2 & 0 \\ 0 & 0 & x_1^2 + x_2^2 + x_3^2 + x_4^2 \end{bmatrix} \quad (6.37)$$

Thus (6.36) represents a 4×3 real orthogonal design. Here $K = 4$, $T = 4$ and $N = 3$. The rate of this real orthogonal design is $R = K/T = 1$. Hence, Eq. (6.36) gives a full rate, real orthogonal design. This can be used for a system with $N = 3$ transmit antennas. It uses $T = 4$ time periods to transmit different symbols from the different antennas as follows

	Antenna 1	Antenna 2	Antenna 3
Time period 1	x_1	x_2	x_3
Time period 2	$-x_2$	x_1	$-x_4$
Time period 3	$-x_3$	x_4	x_1
Time period 4	$-x_4$	$-x_3$	x_2

Since it sends 4 symbols (x_1, x_2, x_3, x_4) using 4 time slots, the rate is unity.

It can be shown that for any number of transmit antennas, N , there exists a full rate, $R = 1$, real space time block code with a block size $T = \min[2^{4c+d}]$, where the minimisation is over all possible integer values of c and d in the set $\{c \geq 0, d \geq 0 | 8c + 2^d \geq N\}$.

Example 6.6 Consider the following 8×7 matrix for $N = 7$ transmit antennas.

$$\mathbf{G} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ -x_2 & x_1 & -x_4 & x_3 & -x_6 & x_5 & x_8 \\ -x_3 & x_4 & x_1 & -x_2 & x_7 & x_8 & -x_5 \\ -x_4 & -x_3 & x_2 & x_1 & x_8 & -x_7 & x_6 \\ -x_5 & x_6 & -x_7 & -x_8 & x_1 & -x_2 & x_3 \\ -x_6 & -x_5 & -x_8 & x_7 & x_2 & x_1 & -x_4 \\ -x_7 & -x_8 & x_5 & -x_6 & -x_3 & x_4 & x_1 \\ -x_8 & x_7 & x_6 & x_5 & -x_4 & -x_3 & -x_2 \end{bmatrix} \quad (6.38)$$

Computing $\mathbf{G}^T \mathbf{G}$ gives

$$\mathbf{G}^T \mathbf{G} = \begin{bmatrix} \sum_{i=1}^8 x_i^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sum_{i=1}^8 x_i^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sum_{i=1}^8 x_i^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sum_{i=1}^8 x_i^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sum_{i=1}^8 x_i^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sum_{i=1}^8 x_i^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sum_{i=1}^8 x_i^2 \end{bmatrix} \quad (6.39)$$

Thus (6.38) represents an 8×7 real orthogonal design. Here $K = 8$, $T = \min[2^{4c+d}] = 8$ for integers $c = 0$ and $d = 3$, and $8c + 2^d = 8 \geq N$.

The rate of this real orthogonal design is $R = K/T = 1$. Hence, Eq. (6.38) represents a full rate, real orthogonal design which can be used for $N = 7$ transmit antennas.

Some interesting observations are given below.

- (i) Orthogonal designs are not unique. Multiplying any orthogonal design \mathbf{G} with a matrix \mathbf{U} having the property $\mathbf{U}^T \mathbf{U} = \mathbf{I}$ results in another orthogonal design $\mathbf{G}' = \mathbf{U}\mathbf{G}$.
- (ii) Deleting a column from an orthogonal design leads to another orthogonal design that can be used to design a STBC with one less transmit antenna. This is called **shortening**.
- (iii) If the original real orthogonal design is delay optimal then the shortened design is also delay optimal.

6.6 Complex Orthogonal Design

LO 2

In this section we will extend the concepts of real orthogonal design to complex orthogonal design.

Definition 6.9 A **Complex Orthogonal Design** of size N is an $N \times N$ matrix G with entries consisting of complex elements drawn from $\pm x_1, \pm x_2, \dots, \pm x_N$ and their conjugates $\pm x_1^*, \pm x_2^*, \dots, \pm x_N^*$, or multiples of these by $j = \sqrt{-1}$ such that

$$\mathbf{G}^H \mathbf{G} = \left(\sum_{i=1}^N |x_i|^2 \right) \mathbf{I}_N \quad (6.40)$$

where \mathbf{I}_N is an $N \times N$ identity matrix. A complex orthogonal design exists if and only if $N = 2$.

Example 6.7 Consider the following 2×2 matrix for $N = 2$ transmit antennas

$$\mathbf{G} = \begin{bmatrix} x_1 & x_2 \\ -x_2^* & x_1^* \end{bmatrix} \quad (6.41)$$

Since $\mathbf{G}^H \mathbf{G} = (|x_1|^2 + |x_2|^2) \mathbf{I}_2$, it is a complex orthogonal design. In fact, it is the Alamouti code discussed in the beginning of the chapter.

Since, we have only one complex orthogonal design of size 2×2 , it motivates us to explore generalised complex orthogonal designs.

Definition 6.10 A Generalised Complex Orthogonal Design is a $T \times N$ matrix \mathbf{G} with complex elements drawn from $0, \pm x_1, \pm x_1^*, \pm x_2, \pm x_2^*, \dots, \pm x_K, \pm x_K^*$, or multiples of these by $j = \sqrt{-1}$ such that

$$\mathbf{G}^H \mathbf{G} = \kappa \left(\sum_{i=1}^K |x_i|^2 \right) \mathbf{I}_N \quad (6.42)$$

where \mathbf{I}_N is an $N \times N$ identity matrix and κ is a constant.

We note the following points:

- (i) In (6.42), it is possible to have $\kappa = 1$ by appropriately normalising the elements of \mathbf{G} .
- (ii) Multiplying a generalised complex orthogonal design by a unitary matrix leads to another generalised complex orthogonal design, i.e., if $\mathbf{U}^H \mathbf{U} = \mathbf{I}$ then, $\mathbf{G}' = \mathbf{U} \mathbf{G}$ is also a generalised complex orthogonal design.

A generalised complex orthogonal design can be used to generate a rate $R = K/T$. Space time code with N antenna elements. Again, let us assume that the transmission is being carried out using a constellation consisting of 2^b symbols. The steps involved in encoding using a generalised complex orthogonal design are as follows.

- 
1. Pick up a block of Kb bits from the input bit-stream.
 2. Based on these Kb bits, select the K symbols from the constellation: (s_1, s_2, \dots, s_k)
 3. Substitute $x_k \rightarrow s_k$ in the matrix \mathbf{G} and generate the codeword $\mathbf{C} = \mathbf{G}(s_1, s_2, \dots, s_k)$. Note that \mathbf{C} is a matrix of size $T \times N$ and consists of linear combinations of s_1, s_2, \dots, s_k and their conjugates.
 4. At time period $t = 1, 2, \dots, T$, transmit the t^{th} row of \mathbf{C} using the different antenna elements, $n = 1, 2, \dots, N$. Thus, the entry $C_{t,n}$ is transmitted from antenna element n at time period t .
 5. At the end of T time periods, effectively K symbols are transmitted.

A complex space-time block code constructed using a $T \times N$ generalised complex orthogonal design provides a diversity of NM for N transmit antennas and M receive antennas. It also results in separate maximum-likelihood decoding of its symbols. Since there are three independent parameters, the number of transmit antennas (N), the number of symbols (K) and the number of time periods (T), the transmission matrix, \mathbf{G} is sometimes denoted by \mathbf{G}_{NKT} .



It is possible to construct a complex orthogonal design using a real orthogonal design. Consider a rate R real orthogonal design with a transmission matrix, \mathbf{G} of size $T \times N$. Denote the conjugate of \mathbf{G} by \mathbf{G}^ , which is obtained by replacing x_k by x_k^* in \mathbf{G} . We can construct a complex orthogonal design as follows.*

$$\mathbf{G}_c = \begin{bmatrix} \mathbf{G} \\ \mathbf{G}^* \end{bmatrix} \quad (6.43)$$

It can be verified that

$$\mathbf{G}_c^H \mathbf{G}_c = 2 \left(\sum_{i=1}^K |x_i|^2 \right) \mathbf{I}_N \quad (6.44)$$

Example 6.8 Consider the following 8×3 matrix for $N = 3$ transmit antennas.

$$\mathbf{G}_{348} = \begin{bmatrix} x_1 & x_2 & x_3 \\ -x_2 & x_1 & -x_4 \\ -x_3 & x_4 & x_1 \\ -x_4 & -x_3 & x_2 \\ x_1^* & x_2^* & x_3^* \\ -x_2^* & x_1^* & -x_4^* \\ -x_3^* & x_4^* & x_1^* \\ -x_4^* & -x_3^* & x_2^* \end{bmatrix} \quad (6.45)$$

Observe that the complex orthogonal design given in (6.45) has been constructed using the method described by (6.43). Computing $\mathbf{G}^H \mathbf{G}$ gives

$$\mathbf{G}^H \mathbf{G} = \begin{bmatrix} \sum_{i=1}^4 |x_i|^2 & 0 & 0 \\ 0 & \sum_{i=1}^4 |x_i|^2 & 0 \\ 0 & 0 & \sum_{i=1}^4 |x_i|^2 \end{bmatrix} \quad (6.46)$$

Thus (6.45) represents an 8×3 generalised complex orthogonal design. Here $K = 4$ and $T = 8$ yielding a rate $R = K/T = 1/2$.

Example 6.9 Consider the following generalised complex orthogonal design for $N = 4$ transmit antennas.

$$\mathbf{G}_{434} = \begin{bmatrix} x_1 & -x_2^* & -x_3^* & 0 \\ x_2 & x_1^* & 0 & x_3^* \\ x_3 & 0 & x_1^* & -x_2^* \\ 0 & -x_3 & x_2 & x_1 \end{bmatrix} \quad (6.47)$$

Here $K = 3$ and $T = 4$ yielding a rate $R = K/T = \frac{3}{4}$. We also note that this design has zeroes as entries in the transmission matrix, \mathbf{G} , implying that one antenna in each time period does not transmit anything (what would be the implication on the overall power consumption?).

This code can be modified for $N = 3$ antennas by simply dropping one of the columns:

$$\mathbf{G}_{334} = \begin{bmatrix} x_1 & -x_2^* & -x_3^* \\ x_2 & x_1^* & 0 \\ x_3 & 0 & x_1^* \\ 0 & -x_3 & x_2 \end{bmatrix} \quad (6.48)$$

The rate $R = K/T = \frac{3}{4}$, however, we now have a design for 3 transmit antennas. Note that *any one* of the columns can be dropped. Similarly, by deleting another column from \mathbf{G}_{334} we can obtain \mathbf{G}_{234} , which is a generalised complex orthogonal design for $N = 2$ transmit antennas with rate $R = \frac{3}{4}$. For the sake of illustration, if we drop the middle column from \mathbf{G}_{334} , we obtain the following design.

$$\mathbf{G}_{234} = \begin{bmatrix} x_1 & -x_3^* \\ x_2 & 0 \\ x_2 & x_1^* \\ 0 & x_2 \end{bmatrix} \quad (6.49)$$

Each time we reduce a transmit antenna, we pay in terms of diversity, since a generalised complex orthogonal design provides a diversity of NM for N transmit antennas and M receive antennas.

Note that the transmission matrix \mathbf{G}_{434} given by (6.47) is not unique. An alternate design, with $R = \frac{3}{4}$, is given below.

$$\mathbf{G}_{434} = \begin{bmatrix} x_1 & x_2 & x_3 & 0 \\ -x_2^* & x_1^* & 0 & x_3 \\ x_3^* & 0 & -x_1^* & x_2 \\ 0 & x_3^* & -x_2 & -x_1 \end{bmatrix} \quad (6.50)$$

It can be shown that the rate of a generalised complex orthogonal design cannot exceed $R = \frac{3}{4}$ for more than two antennas.

Definition 6.11 A **Diagonally Orthogonal Space-Time Block Code** has a $T \times N$ transmission matrix \mathbf{G} with complex elements drawn from $0, \pm x_1, \pm x_1^*, \pm x_2, \pm x_2^*, \dots, \pm x_K, \pm x_K^*$, or multiples of these by $j = \sqrt{-1}$ such that

$$\mathbf{G}^H \mathbf{G} = \mathbf{D} \quad (6.51)$$

where \mathbf{D} is an $N \times N$ diagonal matrix with the diagonal elements $D_{n,n} = l_{n,1} |x_1|^2 + l_{n,2} |x_2|^2 + \dots + l_{n,K} |x_K|^2$.

Compare this with the definition of complex orthogonal design, given by (6.40). In the case of diagonally orthogonal space-time block codes, the diagonal elements of $\mathbf{G}^H \mathbf{G}$ are the linear combinations of $|x_k|^2$. All the wonderful properties of an orthogonal space-time block code, like full diversity gain and simple

ML decoding, are preserved. The encoding and decoding process for the diagonally orthogonal space-time block codes is identical to that of orthogonal space-time block codes, as discussed earlier.

Example 6.10 Consider the following transmission matrix for a diagonally orthogonal space-time block code.

$$\mathbf{G} = \begin{bmatrix} x_1 & x_2 & x_3 & 0 & x_4 \\ -x_2^* & x_1^* & 0 & x_3 & x_5 \\ x_3^* & 0 & -x_1^* & x_2 & x_6 \\ 0 & x_3^* & -x_2^* & -x_1 & x_7 \\ x_4^* & 0 & 0 & -x_7^* & -x_1^* \\ 0 & x_4^* & 0 & x_6^* & -x_2^* \\ 0 & 0 & x_4^* & x_5^* & -x_3^* \\ 0 & -x_5^* & x_6^* & 0 & x_1 \\ x_5^* & 0 & x_7^* & 0 & x_2 \\ -x_6^* & -x_7^* & 0 & 0 & x_3 \\ x_7 & -x_6 & -x_5 & x_4 & 0 \end{bmatrix} \quad (6.52)$$

In this example, $N=5$, $K=7$ and $T=11$, yielding a rate $R=K/T=7/11$. Carrying out the multiplication $\mathbf{G}^H\mathbf{G}$ gives the following diagonal matrix.

$$\mathbf{G}^H\mathbf{G} = \begin{bmatrix} \sum_{i=1}^7 |x_i|^2 & 0 & 0 & 0 & 0 \\ 0 & \sum_{i=1}^7 |x_i|^2 & 0 & 0 & 0 \\ 0 & 0 & \sum_{i=1}^7 |x_i|^2 & 0 & 0 \\ 0 & 0 & 0 & \sum_{i=1}^7 |x_i|^2 & 0 \\ 0 & 0 & 0 & 0 & 2\sum_{i=1}^3 |x_i|^2 + \sum_{i=4}^7 |x_i|^2 \end{bmatrix} \quad (6.53)$$

6.7 Quasi-Orthogonal Space-Time Block Code

LO 2

The two main properties of an orthogonal design are full diversity and simple separate decoding. However, as discussed in the previous section, full-rate complex orthogonal designs do not exist for $N > 2$. Can we relax the simple separate decoding constraint and gain in terms of the rate of the code? This is done in a new class of codes called the **Quasi-orthogonal space-time block codes (QOSTBCs)**. In QOSTBCs, pairs of symbols

can be decoded independently, as opposed to simple separate decoding of individual symbols in orthogonal designs.

Example 6.11 Consider the following QOSTBC.

$$\mathbf{G} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ -x_2^* & x_1^* & -x_4^* & x_3^* \\ x_1 & x_2 & -x_3 & -x_4 \\ -x_2^* & x_1^* & x_4^* & -x_3^* \end{bmatrix} \quad (6.54)$$

Carrying out the multiplication $\mathbf{G}^H \mathbf{G}$ gives

$$\mathbf{G}^H \mathbf{G} = \begin{bmatrix} |x_1|^2 + |x_2|^2 & 0 & 0 & 0 \\ 0 & |x_1|^2 + |x_2|^2 & 0 & 0 \\ 0 & 0 & |x_3|^2 + |x_4|^2 & 0 \\ 0 & 0 & 0 & |x_3|^2 + |x_4|^2 \end{bmatrix} \quad (6.55)$$

Note that this QOSTBC is a full rate code ($R = 1$). It sends four symbols in four time periods. Even though it has four antennas, the diversity provided by this code is only 2 (can you arrive at this conclusion by observing (6.55)?). Note that the first two antennas always transmit (x_1, x_2) while the remaining two always transmit (x_3, x_4) . In fact, this scheme is merely using two Alamouti codes in parallel, two times in a row. So the rate, $R = 1$, is preserved, and the diversity remains 2. To achieve full diversity, instead of substituting $x_k \rightarrow s_k$ in the transmission matrix, we can use the following substitutions to construct the codeword matrix, \mathbf{C} :

$$\begin{aligned} x_1 &= s_1 + s_2, \\ x_2 &= s_3 + s_4, \\ x_3 &= s_1 - s_2, \\ x_4 &= s_3 - s_4. \end{aligned} \quad (6.56)$$

This results in

$$\mathbf{C}^H \mathbf{C} = \begin{bmatrix} a+b & 0 & 0 & 0 \\ 0 & a+b & 0 & 0 \\ 0 & 0 & a-b & 0 \\ 0 & 0 & 0 & a-b \end{bmatrix} \quad (6.57)$$

where $a = \sum_{i=1}^4 |s_i|^2$ and $b = 2 \operatorname{Re}(s_1^* s_2 + s_3^* s_4)$.

Example 6.12 Consider the following QOSTBC.

$$\mathbf{G} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ -x_2^* & x_1^* & -x_4^* & x_3^* \\ -x_3^* & -x_4^* & x_1^* & x_2^* \\ x_4 & -x_3 & -x_2 & x_1 \end{bmatrix} \quad (6.58)$$

This can be rewritten as follows.

$$\mathbf{G} = \begin{bmatrix} G_A(x_1, x_2) & G_A(x_3, x_4) \\ -G_A^*(x_3, x_4) & G_A^*(x_1, x_2) \end{bmatrix} \quad (6.59)$$

where $\mathbf{G}_A(x_1, x_2) = \begin{bmatrix} x_1 & x_2 \\ -x_2^* & x_1^* \end{bmatrix}$ is the standard Alamouti code. Let us denote the i^{th} column of \mathbf{G} by χ_i . Then we have

$$\begin{aligned} <\chi_1, \chi_2> &= 0 \\ <\chi_1, \chi_3> &= 0 \\ <\chi_2, \chi_4> &= 0 \\ <\chi_3, \chi_4> &= 0 \end{aligned} \quad (6.60)$$

where $<\chi_i, \chi_j>$ represents the inner product of the vectors χ_i and χ_j . One way to look at this is that the subspace created by χ_1 and χ_4 is orthogonal to the subspace created by χ_2 and χ_3 . Hence, the name quasi-orthogonal codes.

Example 6.13 Consider the following 4×3 QOSTBC.

$$\mathbf{G} = \begin{bmatrix} x_1 & x_2 & x_3 \\ -x_2^* & x_1^* & -x_4^* \\ -x_3^* & -x_4^* & x_1^* \\ x_4 & -x_3 & -x_2 \end{bmatrix} \quad (6.61)$$

Observe that this QOSTBC has been obtained by dropping the last column of the QOSTBC given by (6.58). This design is for $N=3$ transmit antennas. Using this QOSTBC, we can send four symbols in four time periods. Hence, the rate of this QOSTBC is unity.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 2:

- Consider the code given by $\mathbf{G} = \begin{bmatrix} x_1 & -x_2^* & x_3^* & 0 \\ x_2 & x_1^* & 0 & x_3^* \\ x_3 & 0 & -x_1^* & -x_2^* \\ 0 & x_3 & x_2 & -x_1 \end{bmatrix}$.

M
S
S

- (i) Is it orthogonal?
- (ii) What is the value of N , K and T ?
- (iii) What is the rate of this code?

2. Consider the code given by $\mathbf{G} = \begin{bmatrix} x_1 & x_3 & x_2 \\ -x_2 & -x_4 & x_1 \\ -x_3 & x_1 & x_4 \\ -x_4 & x_2 & -x_3 \end{bmatrix}$.

- (i) Is it orthogonal?
- (ii) What is the value of N and T ?
- (iii) Is this delay optimal?

3. Consider the code given by $\mathbf{G} = \begin{bmatrix} \alpha_1(x_1 + x_2\theta_1) & \alpha_1(x_3 + x_4\theta_1) \\ j\alpha_2(x_3 + x_4\theta_2) & \alpha_2(x_1 + x_2\theta_2) \end{bmatrix}$ where $\theta_1 = \frac{1+\sqrt{5}}{2}$, $\theta_2 = \frac{1-\sqrt{5}}{2}$, $\alpha_1 = 1+j-j\theta_1$ and $\alpha_2 = 1+j-j\theta_2$. Is it an orthogonal STBC? Is it full rate?

4. Design a full-rate STBC scheme using an 8-PSK modulation and fewer than 8 transmit antennas. What will be the transmitted symbols from each antenna corresponding to the input bit-stream 1 1 1 0 1 0 1 0 0 0 1 1 1 1 1 0 1 0 1 0 0 0 1 1 ...?

If you have successfully solved the above problems,
you have mastered LO 2. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/608>



M
S
S

D

6.8 STBC Design Targets and Performance

We can summarise the STBC design targets as follows.



- Rate of transmission
- Diversity gain (rank criterion)
- Multiplexing gain
- Coding gain (determinant criterion)
- Diversity-Multiplexing gain tradeoff (non-vanishing determinant)
- Decoding Complexity

LO 3

Explain the space-time block code design targets and carry out performance analysis.

BER performance curves are presented for various STBCs in Fig. 6.5

for transmission rate equal to 1 b/s/Hz. The channel is assumed to be quasi-static Rayleigh fading channel. ML decoding is used at the receiver. The plots are given for $N = 1, 2, 3$ and 4 transmit antennas and $M = 1$

receive antenna. The choice of the signal constellation and the STBC is listed in Table 6.1. For $N = 4$ using QPSK, the code matrix is given in (6.62).

$$\mathbf{G}_{448} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ -x_2 & x_1 & -x_4 & x_3 \\ -x_3 & x_4 & x_1 & -x_2 \\ -x_4 & -x_3 & x_2 & x_1 \\ * & * & * & * \\ x_1 & x_2 & x_3 & x_4 \\ -x_2^* & x_1^* & -x_4^* & x_3^* \\ -x_3^* & x_4^* & x_1^* & -x_2^* \\ -x_4^* & -x_3^* & x_2^* & x_1^* \end{bmatrix} \quad (6.62)$$

All other STBCs have been discussed earlier. It is interesting to note that the BER performance for $R = 1$, BPSK is identical to $R = \frac{1}{2}$, QPSK for $N = 3$ and 4. We also observe the ‘law of diminishing returns’ in Fig. 6.5. We achieve the maximum gain when we go from $N = 1$ to $N = 2$ (at $\text{BER} = 10^{-4}$ it is approximately 20 dB). Upon going from $N = 2$ to $N = 3$, there is a gain, but not so much (at $\text{BER} = 10^{-4}$ it is approximately 4 dB). Comparing $N = 3$ and $N = 4$, we observe a still smaller gain (at $\text{BER} = 10^{-4}$ it is approximately 2 dB). Also note that the diversity gain increases at higher values of the SNR. For example, compare the curves for $N = 3$ and $N = 4$ at $\text{BER} = 10^{-4}$ and $\text{BER} = 10^{-6}$. The SNR gain almost doubles from 2 dB (at 10^{-4}) to 4 dB (at 10^{-6}).

Figure 6.6 gives the BER curves for $N = 1, 2, 3$ and 4 transmit antennas and $M = 2$ receive antennas. The higher slopes of the curves indicate the increased diversity gain. However, the relative gains have reduced for $M = 2$. Again, let us compare the SNR gain obtained for $N = 3$ and $N = 4$ at $\text{BER} = 10^{-6}$ for $M = 1$ and 2. The SNR gain diminishes from 4 dB ($M = 1$) to 1.5 dB ($M = 2$).

Table 6.1 Choice of the signal constellations and the STBCs

N	<i>Modulation Scheme</i>	<i>Type of Constellation</i>	<i>STBC</i>	<i>Rate, R</i>
1	BPSK	Real	None	Not applicable
2	QPSK	Complex	COD, Alamouti Eq. (6.1)	1
3	BPSK	Real	GROD Eq. (6.36)	1
3	QPSK	Complex	GCOD, \mathbf{G}_{348} Eq. (6.45)	1/2
4	BPSK	Real	ROD Eq. (6.32)	1
4	QPSK	Complex	GCOD, \mathbf{G}_{448} Eq. (6.62)	1/2

ROD = Real Orthogonal Design

COD = Complex Orthogonal Design

GROD = Generalised Real Orthogonal Design,

GCOD = Generalised Complex Orthogonal Design

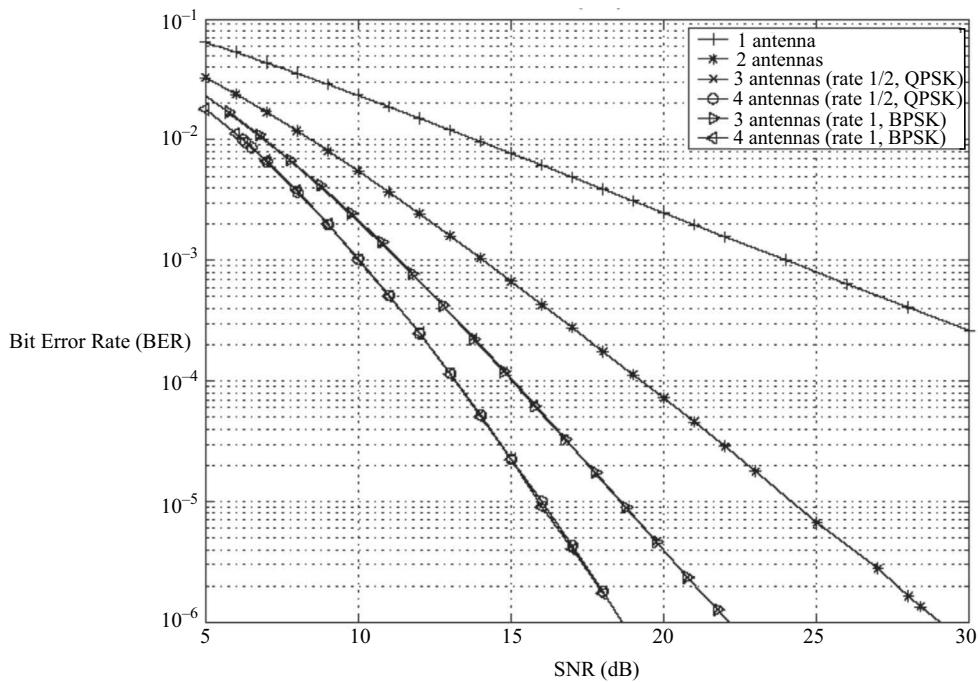


Fig. 6.5 Bit error rate versus SNR for orthogonal STBCs with $M = 1$ receive antenna.

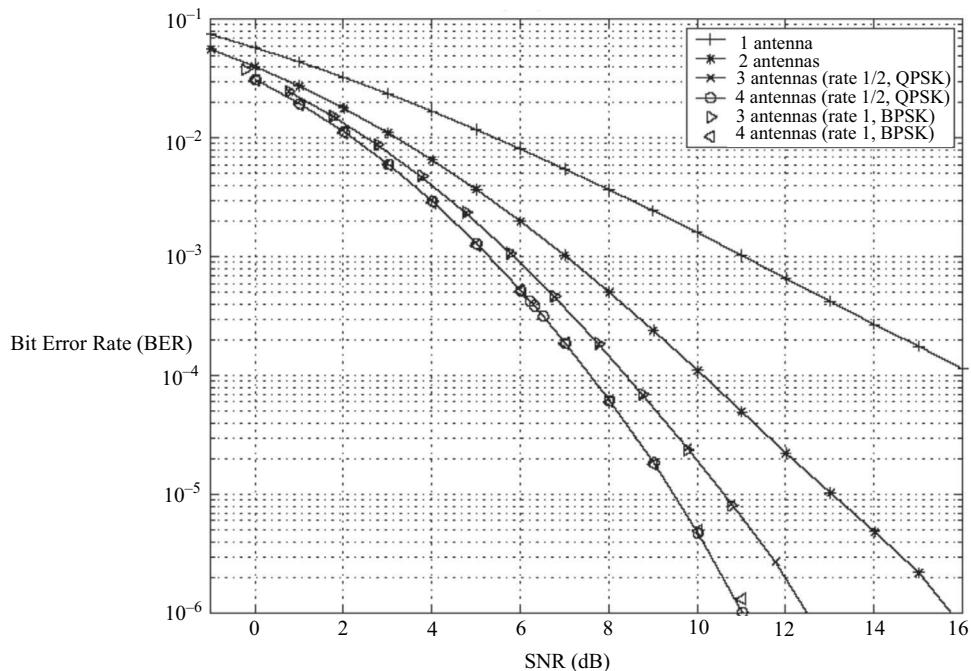


Fig. 6.6 Bit error rate versus SNR for orthogonal STBCs with $M = 2$ receive antennas.

INDUSTRIAL RELEVANCE



Example 6.14 The IEEE 802.11ac is a Wireless Local Area Networks (WLANs) standard that uses STBC. The use of STBC (along with other factors) results in data rates above 1 Gbps. The IEEE 802.11 ac standard can be considered the next step after the IEEE 802.11n standard. The IEEE 802.11ac defines the core 2×1 , 4×2 , 6×3 , and 8×4 STBC modes. The 2×1 , access-point to client, is the most commonly used configuration.

Example 6.15 Underwater acoustic channels are one of the most challenging communication media. Acoustic propagation is characterised by two important factors: attenuation that increases with signal frequency and time-varying multipath propagation leading to fading. To overcome the detrimental effects of fading, diversity gain offered by STBC is an obvious choice. Therefore, STBCs find utility in underwater acoustic communications. OFDM-MIMO systems are used in underwater acoustic communications for spatial diversity gain. Engineers have also tried Alamouti space-frequency block coding over the carriers of an orthogonal frequency division multiple access system in underwater communication.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 3:

- Plot bit error rate versus SNR curves for orthogonal STBCs for $N = 1, 2, 3, 4$ transmit antennas and $M = 1$ receive antenna. Determine the diversity gain using
$$g_d = -\lim_{\text{SNR} \rightarrow \infty} \frac{\log_2(P_e)}{\log_2(\text{SNR})}$$
. Measure the slope of each curve and comment on the diversity gain. At $P_e \approx 10^{-4}$, how much less transmit power is required for $N = 2$, versus $N = 4$? M
- Plot bit error rate versus SNR curves for orthogonal STBCs for $N = 1, 2, 3, 4$ transmit antennas and $M = 2$ receive antennas. Measure the slope of each curve and comment on the diversity gain. At $P_e \approx 10^{-4}$, how much less transmit power is required for $N = 2$, versus a SISO system? M

If you have successfully solved the above problems,
you have mastered LO 3. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/609>



6.9 Concluding Remarks

One of the early bandwidth-efficient transmit antenna diversity schemes was proposed by Wittneben in 1993. It was followed by the work of N. Seshadri and J. H. Winters, who used transmitter antenna diversity for frequency-division-duplex (FDD) transmission systems. Multilayered space-time architecture was introduced by G.J. Foschini in 1996. Space-time codes were proposed by V. Tarokh, N. Seshadri and A. R. Calderbank in 1998, and in the same year the Alamouti code was introduced. Further work was carried out by V. Tarokh and H. Jafarkhani. In 2003, J. N. Laneman and G. W. Wornell carried out initial work on distributed space-time coding, which exploits cooperative diversity in wireless networks. The use of MIMO technologies is one of the important distinctions between 3G and 4G wireless technologies. Both space-time codes and distributed space-time codes have been used in many newer wireless standards. For example, WiMax and LTE have adopted the frequency domain version of the Alamouti code.

LEARNING OUTCOMES

- The probability of erroneously decoding codeword \mathbf{C}^2 when codeword \mathbf{C}^1 was transmitted is called the pairwise error probability, and is denoted by $P(\mathbf{C}^1 \rightarrow \mathbf{C}^2)$.
- $\mathbf{A}(\mathbf{C}^1, \mathbf{C}^2)$, defined as $(\mathbf{C}^2 - \mathbf{C}^1)^H (\mathbf{C}^2 - \mathbf{C}^1) = D(\mathbf{C}^2, \mathbf{C}^1)^H D(\mathbf{C}^2, \mathbf{C}^1)$.
- Rank criteria: In order to achieve maximum diversity, the matrix $\mathbf{A}(\mathbf{C}^i, \mathbf{C}^j)$ should be of full rank for any two codewords, $\mathbf{C}^i \neq \mathbf{C}^j$. The smallest value of r , over any pair of codewords, provides a diversity gain of rM .
- Determinant criteria: In order to achieve maximum coding gain, the minimum determinant of the matrix $\mathbf{A}(\mathbf{C}^i, \mathbf{C}^j)$ should be maximised for any two codewords, $\mathbf{C}^i \neq \mathbf{C}^j$.
- The Alamouti code is given by the matrix $\mathbf{X} = \begin{bmatrix} x_1 & x_2 \\ -x_2^* & x_1^* \end{bmatrix}$.
- Orthogonal Space-time block codes provide (i) simple decoding and (ii) maximum diversity.
- A real orthogonal design of size N is an $N \times N$ matrix $\mathbf{G} = \mathbf{G}(x_1, x_2, \dots, x_N)$, with entries consisting of real elements drawn from $\pm x_1, \pm x_2, \dots, \pm x_N$ such that $\mathbf{G}^T \mathbf{G} = \left(\sum_{i=1}^N x_i^2 \right) \mathbf{I}_N$ where \mathbf{I}_N is a $N \times N$ identity matrix.
- A real orthogonal design exists if and only if $N = 2, 4$ and 8 .
- A generalised real orthogonal design is a $T \times N$ matrix with entries consisting of real elements drawn from $\pm x_1, \pm x_2, \dots, \pm x_K$ such that $\mathbf{G}^T \mathbf{G} = \left(\sum_{i=1}^K x_i^2 \right) \mathbf{I}_N$ where \mathbf{I}_N is a $N \times N$ identity matrix and K is a constant that represents the number of distinct symbols being transmitted.
- The rate \mathbf{G} of a generalised real orthogonal is defined as $R = \frac{K}{T}$.
- A generalised real orthogonal design with rate $R = 1$ is called a full rate orthogonal design.
- An orthogonal design with minimum possible value of the blocklength T is called delay optimal.
- Orthogonal designs are not unique. Multiplying any orthogonal design \mathbf{G} with a matrix \mathbf{U} having the property $\mathbf{U}^T \mathbf{U} = \mathbf{I}$ results in another orthogonal design $\mathbf{G}' = \mathbf{UG}$.

- Deleting a column from an orthogonal design leads to another orthogonal design that can be used to design a STBC with one less transmit antenna. This is called shortening.
- If the original real orthogonal design is delay optimal then the shortened design is also delay optimal.
- A complex orthogonal design of size N is an $N \times N$ matrix \mathbf{G} with entries consisting of complex elements drawn from $\pm x_1, \pm x_2, \dots, \pm x_N$ and their conjugates $\pm x_1^*, \pm x_2^*, \dots, \pm x_N^*$, or multiples of these by $j = \sqrt{-1}$ such that $\mathbf{G}^H \mathbf{G} = \left(\sum_{i=1}^N |x_i|^2 \right) \mathbf{I}_N$ where \mathbf{I}_N is a $N \times N$ identity matrix.
- A complex orthogonal design exists if and only if $N = 2$.
- A generalised complex orthogonal design is a $T \times N$ matrix \mathbf{G} with complex elements drawn from $0, \pm x_1, \pm x_1^*, \pm x_2, \pm x_2^*, \dots, \pm x_K, \pm x_K^*$, or multiples of these by $j = \sqrt{-1}$ such that $\mathbf{G}^H \mathbf{G} = \kappa \left(\sum_{i=1}^K |x_i|^2 \right) \mathbf{I}_N$ where \mathbf{I}_N is a $N \times N$ identity matrix and κ is a constant.
- Multiplying a generalised complex orthogonal design by a unitary matrix leads to another generalised complex orthogonal design, that is, if $\mathbf{U}^H \mathbf{U} = \mathbf{I}$ then, $\mathbf{G}' = \mathbf{U} \mathbf{G}$ is also a generalised complex orthogonal design.
- Since there are three independent parameters, the number of transmit antennas (N), the number of symbols (K) and the number of time periods (T), the transmission matrix, \mathbf{G} is sometimes denoted by \mathbf{G}_{NKT} .
- A diagonally orthogonal space-time block code has a $T \times N$ transmission matrix \mathbf{G} with complex elements drawn from $0, \pm x_1, \pm x_1^*, \pm x_2, \pm x_2^*, \dots, \pm x_K, \pm x_K^*$, or multiples of these by $j = \sqrt{-1}$ such that $\mathbf{G}^H \mathbf{G} = \mathbf{D}$ where \mathbf{D} is an $N \times N$ diagonal matrix with the diagonal elements $D_{n,n} = l_{n,1} |x_1|^2 + l_{n,2} |x_2|^2 + \dots + l_{n,K} |x_K|^2$.

The essence of mathematics is its freedom.

Georg Cantor (1845-1918)



MULTIPLE CHOICE QUESTIONS

6.1 $A(\mathbf{C}^1, \mathbf{C}^2)$ is defined as

- | | |
|---|---|
| (a) $(\mathbf{C}^2 - \mathbf{C}^1)^H (\mathbf{C}^2 - \mathbf{C}^1)$ | (b) $D(\mathbf{C}^2, \mathbf{C}^1)^H D(\mathbf{C}^2, \mathbf{C}^1)$ |
| (c) Both (a) and (b) | (d) None of the above |

6.2 Rank criteria suggests that in order to achieve maximum diversity

- The matrix $A(\mathbf{C}^i, \mathbf{C}^j)$ should be of full rank for any two codewords, $\mathbf{C}^i \neq \mathbf{C}^j$
- The matrix $A(\mathbf{C}^i, \mathbf{C}^j)$ should be orthogonal for any two codewords, $\mathbf{C}^i \neq \mathbf{C}^j$

- (c) The matrix $\mathbf{A}(\mathbf{C}^i, \mathbf{C}^j)$ should be unitary for any two codewords, $\mathbf{C}^i \neq \mathbf{C}^j$
 (d) None of the above

6.3 Determinant criteria suggests that in order to achieve maximum coding gain

- (a) The maximum determinant of the matrix $\mathbf{A}(\mathbf{C}^i, \mathbf{C}^j)$ should be minimised for any two codewords, $\mathbf{C}^i \neq \mathbf{C}^j$
 (b) The minimum determinant of the matrix $\mathbf{A}(\mathbf{C}^i, \mathbf{C}^j)$ should be maximised for any two codewords, $\mathbf{C}^i \neq \mathbf{C}^j$
 (c) The minimum determinant of the matrix $\mathbf{A}(\mathbf{C}^i, \mathbf{C}^j)$ should be minimised for any two codewords, $\mathbf{C}^i \neq \mathbf{C}^j$
 (d) None of the above

6.4 The Alamouti code is given by

$$(a) \quad \mathbf{X} = \begin{bmatrix} x_1 & x_2 \\ x_2^* & x_1^* \end{bmatrix}$$

$$(b) \quad \mathbf{X} = \begin{bmatrix} x_1 & x_2 \\ -x_2^* & x_1 \end{bmatrix}$$

$$(c) \quad \mathbf{X} = \begin{bmatrix} x_1 & x_2 \\ -x_2 & x_1^* \end{bmatrix}$$

$$(d) \quad \mathbf{X} = \begin{bmatrix} x_1 & x_2 \\ -x_2^* & x_1^* \end{bmatrix}$$

6.5 Orthogonal space-time block codes provide

- (a) Simple decoding (b) Maximum diversity
 (c) Both (a) and (b) (d) None of the above

6.6 A real orthogonal design of size N is an $N \times N$ such that

$$(a) \quad \mathbf{G}^T \mathbf{G} = \left(\sum_{i=1}^N x_i \right) \mathbf{I}_N$$

$$(b) \quad \mathbf{G}^T \mathbf{G} = \left(\sum_{i=1}^N |x_i| \right) \mathbf{I}_N$$

$$(c) \quad \mathbf{G}^T \mathbf{G} = \left(\sum_{i=1}^N x_i^2 \right) \mathbf{I}_N$$

(d) None of the above

6.7 A real orthogonal design exists if, and only if, N is equal to

- (a) 2 (b) 4
 (c) 8 (d) All of the above

6.8 The rate \mathbf{G} of a generalised real orthogonal is defined as

$$(a) \quad R = \frac{K}{N}$$

$$(b) \quad R = \frac{T}{K}$$

$$(c) \quad R = \frac{K}{T}$$

(d) None of the above

6.9 Orthogonal designs

- (a) Are unique (b) Are not unique
 (c) Always have rate $R = 1$ (d) None of the above

6.10 Multiplying a generalised complex orthogonal design by a unitary matrix leads to

- (a) Increase in the rate, R
 (b) A quasi-orthogonal design

- (c) Another generalised complex orthogonal design
 (d) None of the above

For interactive quiz with answers, scan the QR code given here

Or

Visit <http://qrcode.flipick.com/index.php/571>



SHORT-ANSWER TYPE QUESTIONS



- 6.1 A real orthogonal design exists if, and only if, $N = 2$. True or false?
- 6.2 What is the rate of the Alamouti code?
- 6.3 What is the diversity provided by the Alamouti code?
- 6.4 In STBC, what is the probability of erroneously decoding codeword C^2 when codeword C^1 was transmitted known as?
- 6.5 Give the diversity gain of a STBC in terms of the rank of the matrix $A(C^1, C^2)$ and the number of receive antennas.
- 6.6 Give the coding gain of a STBC in terms of the matrix $A(C^1, C^2)$.
- 6.7 Under what condition is full diversity of an STBC achieved?
- 6.8 Give a real orthogonal design for $N = 2$.
- 6.9 Define delay optimal generalised orthogonal design.
- 6.10 Give a 4×3 generalised real orthogonal design.
- 6.11 Define shortening in the context of STBC.
- 6.12 Multiplying a generalised complex orthogonal design by a unitary matrix leads to another generalised complex orthogonal design. True or false?
- 6.13 Is it possible to construct a complex orthogonal design using a real orthogonal design?

6.14 For $\mathbf{G}_{434} = \begin{bmatrix} x_1 & -x_2^* & -x_3^* & 0 \\ x_2 & x_1^* & 0 & x_3^* \\ x_3 & 0 & x_1^* & -x_2^* \\ 0 & -x_3 & x_2 & x_1 \end{bmatrix}$ determine the parameters K , T and R .

6.15 Is $\mathbf{G} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ -x_2^* & x_1^* & -x_4^* & x_3^* \\ -x_3^* & -x_4^* & x_1^* & x_2^* \\ x_4 & -x_3 & -x_2 & x_1 \end{bmatrix}$ an orthogonal design?

For answers, scan the QR code given here

Or

Visit <http://qrcode.flipick.com/index.php/610>





PROBLEMS

- M** 6.1 Suppose we have a 2×1 wireless system that employs orthogonal STBC and uses QPSK modulation. Let the input bit-stream be 1 1 1 1 0 0 1 0 1 1 1 0 ... What will be the transmitted symbols from each antenna?
- M** 6.2 Verify whether the following is a full rate, real orthogonal design.

$$\mathbf{G} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \\ -x_2 & x_1 & -x_4 & x_3 & -x_6 \\ -x_3 & x_4 & x_1 & -x_2 & x_7 \\ -x_4 & -x_3 & x_2 & x_1 & x_8 \\ -x_5 & x_6 & -x_7 & -x_8 & x_1 \\ -x_6 & -x_5 & -x_8 & x_7 & x_2 \\ -x_7 & -x_8 & x_5 & -x_6 & -x_3 \\ -x_8 & x_7 & x_6 & x_5 & -x_4 \end{bmatrix} \quad (6.63)$$

- M** 6.3 Verify whether the following is a full rate, real orthogonal design.

$$\mathbf{G} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \\ -x_2 & x_1 & -x_4 & x_3 & -x_6 \\ -x_3 & -x_4 & x_1 & -x_2 & -x_7 \\ -x_4 & x_3 & -x_2 & -x_1 & x_8 \\ -x_5 & x_6 & x_7 & x_8 & x_1 \\ -x_6 & x_5 & -x_8 & x_7 & -x_2 \\ -x_7 & -x_8 & x_5 & -x_6 & x_3 \\ -x_8 & -x_7 & x_6 & x_5 & x_4 \end{bmatrix} \quad (6.64)$$

- M** 6.4 Verify whether the following is a full rate, real orthogonal design.

$$\mathbf{G} = \begin{bmatrix} x_1 & x_2 & x_3 \\ x_2 & x_1 & -x_4 \\ x_3 & -x_4 & x_5 \\ x_4 & x_5 & -x_1 \\ x_5 & -x_3 & x_2 \end{bmatrix} \quad (6.65)$$

- S** 6.5 Show that multiplying any real orthogonal design \mathbf{G} with a matrix \mathbf{U} having the property $\mathbf{U}^T \mathbf{U} = \mathbf{I}$ results in another real orthogonal design $\mathbf{G}' = \mathbf{U}\mathbf{G}$.
- D** 6.6 Does a 16×9 real orthogonal design exist? If yes, give the design.
- M** 6.7 Show that deleting a column from an orthogonal design leads to another orthogonal design.
- M** 6.8 (i) Show that if the original orthogonal design is delay optimal then the shortened design is also delay optimal.

(ii) Give all possible shortened designs starting from the real orthogonal design given in (6.32).

- S** 6.9 If \mathbf{G} represents a generalised complex orthogonal design, show that $\mathbf{G}' = \mathbf{U}\mathbf{G}$ is also a generalised complex orthogonal design, where \mathbf{U} is a unitary matrix ($\mathbf{U}^H\mathbf{U} = \mathbf{I}$).
- M** 6.10 Verify whether the following is a generalised complex orthogonal design.

$$\mathbf{G} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ -x_2 & x_1 & -x_4 & x_3 \\ -x_3 & x_4 & x_1 & -x_2 \\ -x_4 & -x_3 & x_2 & x_1 \\ x_1^* & x_2^* & x_3^* & x_4^* \\ -x_2^* & x_1^* & -x_4^* & x_3^* \\ -x_3^* & x_4^* & x_1^* & -x_2^* \\ -x_4^* & -x_3^* & x_2^* & x_1^* \end{bmatrix} \quad (6.66)$$

What is the rate of this code?

- D** 6.11 Give the generalised complex orthogonal design for the following parameters: $K = 8$, $T = 16$ and $N = 8, 7, 6$ and 5 .
- M** 6.12 Consider a 4×1 wireless MIMO system that employs the QOSTBC given in (6.54). The encoder uses QPSK modulation. What are the transmitted symbols for the following input bit-stream: $0\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ \dots$?
- M** 6.13 Verify whether the following is a QOSTBC.

$$\mathbf{G} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ -x_2^* & x_1^* & -x_4^* & x_3^* \\ -x_3 & -x_4 & x_1 & x_2 \\ x_4^* & -x_3^* & -x_2^* & x_1^* \end{bmatrix} \quad (6.67)$$

- M** 6.14 Verify whether the following is a QOSTBC:

$$\mathbf{G} = \begin{pmatrix} x_1 & x_2 & x_3 & 0 & x_4 & x_5 & x_6 & 0 \\ -x_2^* & -x_1^* & 0 & -x_3 & x_5^* & -x_4^* & 0 & x_6 \\ x_3^* & 0 & -x_1^* & -x_2 & -x_6^* & 0 & x_4^* & x_5 \\ 0 & -x_3^* & x_2^* & -x_1 & 0 & x_6^* & -x_5^* & x_4 \\ -x_4 & -x_5 & -x_6 & 0 & x_1 & x_2 & x_3 & 0 \\ -x_5^* & x_4^* & 0 & x_6 & -x_2^* & x_1^* & 0 & x_3 \\ x_6^* & 0 & -x_4^* & x_5 & x_3^* & 0 & -x_1^* & x_2 \\ 0 & x_6^* & -x_5^* & -x_4 & 0 & x_3^* & -x_2^* & -x_1 \end{pmatrix} \quad (6.68)$$

What is the rate of this code?

- D** 6.15 (i) Fill in the missing symbols to make the following code a generalised real orthogonal design for a five-antenna system.

$$\mathbf{G} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \\ -x_2 & [] & x_4 & -x_3 & x_6 \\ -x_3 & -x_4 & [] & x_2 & x_7 \\ -x_4 & x_3 & -x_2 & [] & x_8 \\ -x_5 & -x_6 & -x_7 & -x_8 & [] \\ -x_6 & x_5 & [] & x_7 & -x_2 \\ -x_7 & [] & x_5 & -x_6 & -x_3 \\ [] & -x_7 & x_6 & x_5 & -x_4 \end{bmatrix}$$

- D** (ii) Add one additional column to the STBC above in order to make it a 6×8 generalised real orthogonal design for a six-antenna system.

COMPUTER PROBLEMS



- 6.1 Write a computer program to implement the Alamouti code as given in (6.1). Carry out Monte Carlo simulations to determine the BER performance of the code over a Rayleigh fading channel.
- 6.2 Write a computer program to implement the QOSTBC as given in (6.54). Carry out Monte Carlo simulations to determine the BER performance of the code versus SNR over a Rayleigh fading channel.
- 6.3 Generate a 2D mesh plot for the capacity of the MIMO system as a function of the number of transmit antennas, N and the number of receive antennas, M for a given $SNR = 20$ dB. Repeat for $SNR = 25$ and 30 dB. You can vary both N and M between 1 and 20. Comment on your observations.

- 6.4 Write a computer program to implement the code given by $\mathbf{G}_{434} = \begin{bmatrix} x_1 & -x_2^* & -x_3^* & 0 \\ x_2 & x_1^* & 0 & x_3^* \\ x_3 & 0 & x_1^* & -x_2^* \\ 0 & -x_3 & x_2 & x_1 \end{bmatrix}$. Carry

out Monte Carlo simulations to determine the BER performance of the code over a Rayleigh fading channel and a Rician fading channel. In which fading scenario is it more effective? Why?

PROJECT IDEAS

- 6.1 **Combined STBC–LDPC Encoder and Decoder:** The aim is to combine non-LDPC codes in a MIMO channel with a high order Galois field in order to compute the log likelihood ratios (LLRs) for each transmitted symbol. Combine the STBC decoder with the LDPC decoder together to achieve the full diversity in MIMO channels.

- 6.2 **Double-symbol Decodable Design:** Consider the QOSTBC given in (6.58). Write a program that efficiently performs double symbol decoding for this STBC.
- 6.3 **Alternate Double-symbol Decodable Designs:** Consider the QOSTBC given in (6.69). Verify that it is a double-symbol decodable design. Compare it with the design given in (6.58). Write a computer program that intelligently constructs variations of the design that are also double-symbol decodable.

$$\mathbf{G} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ -x_2^* & x_1^* & -x_4^* & x_3^* \\ x_3 & x_4 & x_1 & x_2 \\ -x_4^* & x_3^* & -x_2^* & x_1^* \end{bmatrix} \quad (6.69)$$

- 6.4 **STBC for Free-Space Optical (FSO) Communications:** Consider optical transmission using intensity modulation and direct detection (IM/DD). It is known that atmospheric turbulence degrades the link performance drastically. In FSO communication, typically on-off keying (OOK) is used for data transmission. Design a modified Alamouti code for FSO communications and carry out simulations to plot BER versus SNR. How will you encode the ‘off’ state?

REFERENCES FOR FURTHER READING

1. Jafarkhani, H.; *Space-Time Coding: Theory and Practice*. Cambridge University Press, 2005.
2. Vucetic, B. and J. Yuan; *Space-Time Coding*. Wiley, 2003.

PART - III

Codes on Graph

Chapter 7 Convolutional Codes

Chapter 8 Trellis Coded Modulation

Convolutional Codes

The shortest path between two truths in the real domain passes through the complex domain.

Jacques Hadamard (1865-1963)

Learning objectives

After studying this chapter the students will be able to:

- LO 1** Know the concept of convolutional codes, use polynomial and generator polynomial matrix to describe convolutional codes and carry out encoding using convolutional encoders.
- LO 2** Explain how to perform Viterbi decoding and apply distance bounds. The performance bounds and some known good convolutional codes will also be studied.
- LO 3** Underline the concept of Turbo coding and decoding and also interleaver design for Turbo codes.

7.1 Introduction to Convolutional Codes

So far we have studied block codes, where a block of k information symbols are encoded into a block of n coded symbols. There is always a one-to-one correspondence between the uncoded block of symbols (information word) and the coded block of symbols (codeword). This method is particularly useful for high data rate applications, where the incoming stream of uncoded data is first broken into blocks, encoded and then transmitted (Fig. 7.1). A large blocklength is important because of the following reasons.

...
This chapter comes with a video
overview by the author. Scan here to
know more or
Visit <http://qrcode.flipick.com/index.php/616>



- (i) Many of the good codes that have large distance properties are of large blocklengths (e.g., the RS codes).
- (ii) Larger block lengths imply that the encoding overhead is small.

However, very large blocklengths have the disadvantage that unless the entire block of encoded data is received at the receiver, the decoding procedure cannot start, which may result in delays. In contrast, there is another coding scheme in which much smaller blocks of uncoded data of length k_0 are used. These are called **Information Frames**. An information frame typically contains just a few symbols, and can have as few as just one symbol! These information frames are encoded into **Codeword Frames** of length n_0 . However, just one information frame is not used to obtain the codeword frame. Instead, the current information frame along with earlier m information frames is used to obtain a single codeword frame. This implies that such encoders have *memory*, which retains the previous m incoming information frames. The codes that are obtained in this fashion are called tree codes. An important sub-class of **Tree Codes**, used frequently in practice, is called **Convolutional Codes**. Up till now, all the decoding techniques discussed are algebraic and memoryless, i.e., decoding decisions are based only on the current codeword. Convolutional codes make decisions based on past information, i.e., memory is required. Thus, these are codes with memory (one more trick up our sleeves to outwit noise!).

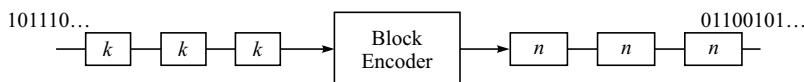


Fig. 7.1 Encoding using a block encoder.

In this chapter

We shall start with an introduction to tree and trellis codes. We will then develop the necessary mathematical tools to construct convolutional codes. We will see that convolutional codes can be easily represented by polynomials. Subsequently, we will introduce the notion of generator polynomial matrices for convolutional codes, in LO 1.

Next, we will study the distance notions for convolutional codes and the generating function. We will then learn how to perform the famous Viterbi decoding. We will look at the upper bounds on the minimum distance of convolutional codes. The performance bounds, along with some known good convolutional codes, will also be studied, in LO 2.

Finally, we shall introduce the concept of Turbo coding and decoding. Interleaver design for Turbo codes will also be discussed, in LO 3.

7.2 Tree Codes and Trellis Codes

LO 1



Know the concept of convolutional codes, use polynomial and generator polynomial matrix to describe convolutional codes and carry out encoding using convolutional encoders.

In this chapter, we assume that there is an infinitely long stream of incoming symbols. Thanks to the large volumes of data to be sent these days, it is not a bad assumption. Today, we have more bits of data than the known number of stars in the universe. By 2020 we might be looking at 50 zettabytes of data ($1 \text{ ZB} = 10^{21} \text{ bytes} = 2^{70} \text{ bytes}$). Big data is here to stay!

Convolutional encoders comprise two basic components which are as follows.

- (i) memory, which is basically a shift register, and
- (ii) a logic circuit.

Example 7.1 Consider a basic memory unit as shown in Fig. 7.2 (a). The ‘D’ could be interpreted as a delay element, or a D-flip-flop, or a simple 1-bit memory. A basic logic circuit could simply be an XOR gate, or a binary adder without carry, as shown in Fig. 7.2 (b).



Fig. 7.2 (a) One memory unit. (b) A binary adder without carry.

Example 7.2 Consider a circuit with m memory units as shown in Fig. 7.3.

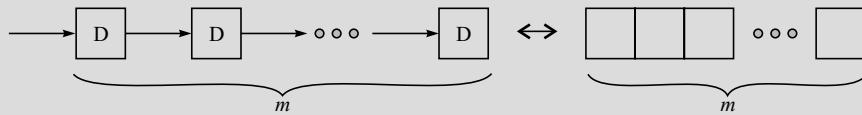


Fig. 7.3 Two ways to represent a circuit with m memory units.

We can represent the encoder memory as concatenated m discrete memory units, or equivalently, by a shift register of length m .

For coding, an infinitely long stream of incoming symbols is first broken up into segments of k_0 symbols, as shown in Fig. 7.4. Each segment is called an information frame. The memory of a convolutional encoder can store m information frames. Each time a new information frame arrives, it is shifted into the shift register and the oldest information frame is discarded. At the end of any frame time the encoder has m most recent information frames in its memory, which corresponds to a total of mk_0 information symbols.

When a new frame arrives, the encoder computes the codeword frame using this new frame that has just arrived and the m frames stored previously. The computation of the codeword frame is done using the logic circuit. This codeword frame is then shifted out. The oldest information frame is the memory, which is then discarded and the most recent information frame is shifted into the memory. The encoder is now ready for the next incoming information frame. Thus for every information frame (k_0 symbols) that comes in, the encoder generates a codeword frame (n_0 symbols). It should be observed that the same information frame may not generate the same codeword frame because the codeword frame also depends on the m previous information frames.

Intuition **Definition 7.1** The **Constraint Length** of a shift register encoder is defined as the number of symbols it can store in its memory. If the shift register encoder stores m previous information frames of length k_0 , the constraint length of this encoder $v = mk_0$.

We shall give a more formal definition of constraint length later in this chapter.

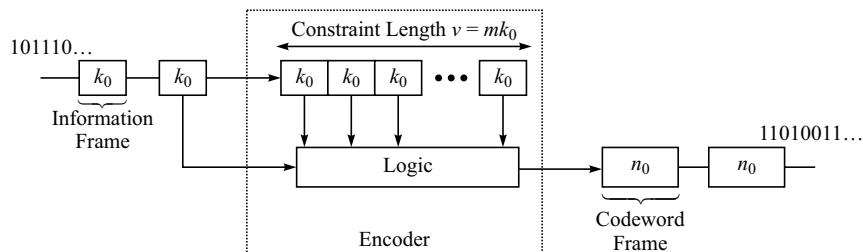


Fig. 7.4 A shift register encoder that generates a tree code.

Definition 7.2 The infinite set of all infinitely long codewords obtained by feeding every possible input sequence to a shift register encoder is called an (n_0, k_0) **Tree Code**. The rate of this tree code is defined as

$$R = \frac{k_0}{n_0} \quad (7.1)$$

A more formal definition is that a (n_0, k_0) **tree code** is a mapping from the set of semi-infinite sequences of elements of $GF(q)$ into itself such that if for any M , two semi-infinite sequences agree in the first Mk_0 components, then their images agree in the first Mn_0 components.

Definition 7.3 The **Word length** of a shift register encoder is defined as $k = (m + 1)k_0$. The **Blocklength** of a shift register encoder is defined as $n = (m + 1)n_0 = k \frac{n_0}{k_0}$.

Note that the code rate $R = \frac{k_0}{n_0} = \frac{k}{n}$. Normally, for practical shift register encoders, the information frame length k_0 is small (usually less than 5). Therefore, it is difficult to obtain the code rate R of tree codes close to unity, as is possible with block codes (e.g., RS codes).

Definition 7.4 An (n_0, k_0) tree code that is linear, time-invariant and has a finite word length $k = (m + 1)k_0$ is called an (n, k) **Convolutional Code**.

Definition 7.5 An (n_0, k_0) tree code that is time-invariant and has a finite word length k is called an (n, k) **Sliding Block Code**. Thus, a linear sliding block code is a convolutional code.

Example 7.3 Consider the convolutional encoder given in Fig. 7.5.

This encoder takes in one bit at a time and encodes it into 2 bits. The information frame length $k_0 = 1$, the codeword frame length $n_0 = 2$ and the blocklength $(m + 1)n_0 = 6$. The constraint length of this encoder $v = 2$. The code rate of this encoder is $\frac{1}{2}$. The clock rate of the outgoing data is twice as fast as that of the incoming data. The adders are binary adders, and from the point of view of circuit implementation, are simply XOR gates. Note that the encoder is inside the box drawn with dotted line. There is a memory unit outside the encoder, which serves as the input buffer. Thus, the convolutional encoder has only 2 memory units.



Let us assume that the initial state of the shift register is [0 0]. Now, either '0' will come or '1' will come as the incoming bit. Suppose '0' comes. On performing the logic operations, we see that the computed value of the codeword frame is [0 0]. The 0 will be pushed into the memory (shift register) and the rightmost '0' will be dropped. The state of the shift register remains [0 0]. Next, let '1' arrive at the encoder. Again we perform the logic operations to compute the codeword frame. This time we obtain [1 1]. So, this will be pushed out as the encoded frame. The

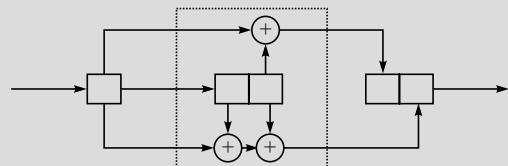


Fig. 7.5 Convolutional encoder with $k_0 = 1$, $n_0 = 2$ and $m = 2$.

incoming ‘1’ will be shifted into the memory, and the rightmost bit will be dropped, as depicted in Fig. 7.6. So the new state of the shift register will be [1 0].



Fig. 7.6 The ‘pushy’ shift register.

Again, there are two possibilities regarding the incoming bit: ‘0’ or ‘1’. It is possible to construct a table which lists all the possibilities (Table 7.1).

Table 7.1 The incoming and outgoing bits of the convolutional encoder given in Example 7.3.

Incoming Bit	Current State of the Encoder		Outgoing Bits	
0	0	0	0	0
1	0	0	1	1
0	0	1	1	1
1	0	1	0	0
0	1	0	0	1
1	1	0	1	0
0	1	1	1	0
1	1	1	0	1

We observe that there are only $2^2 = 4$ possible states of the shift register. So, we can construct the state diagram of the encoder as shown in Fig. 7.7. The bits associated with each arrow represent the incoming bit. It can be seen that the same incoming bit gets encoded *differently* depending on the current state of the encoder. This is different from the linear block codes studied in previous chapters where there is always a one-to-one correspondence between the incoming uncoded block of symbols (information word) and the coded block of symbols (codeword).

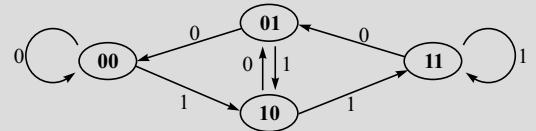


Fig. 7.7 The state diagram for the encoder in Example 7.3.

Example 7.4 Consider the convolutional encoder given in Fig. 7.5. Suppose we wish to encode the following bit stream: 0 1 0 0 1 0 1 1 ...

We assume that the initial state of the encoder is [0 0]. We make use of Table 7.1.

The corresponding output bit stream will be: 00 11 01 11 11 01 00 10 ...

Suppose the input bit stream was 1 1 0 0 1 0.

The corresponding output bit stream will be: 11 10 10 11 11 01.

Example 7.5 Consider the convolutional encoder given in Fig. 7.5. An equivalent circuit representation is given in Fig. 7.8.

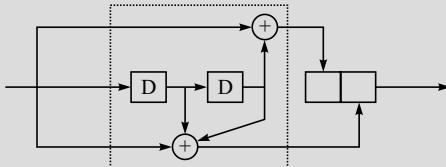


Fig. 7.8 An equivalent representation of the encoder given in Fig. 7.5.

The information contained in the state diagram can be conveyed usefully in terms of a graph called the **Trellis Diagram**. A trellis is a graph whose nodes are in a rectangular grid, which is semi-infinite to the right. Hence, these codes are also called **Trellis Codes**. The number of nodes in each column is finite. The following example gives an illustration of a trellis diagram.

Example 7.6 The trellis diagram for the convolutional encoder discussed in Example 7.3 is given in Fig. 7.9.

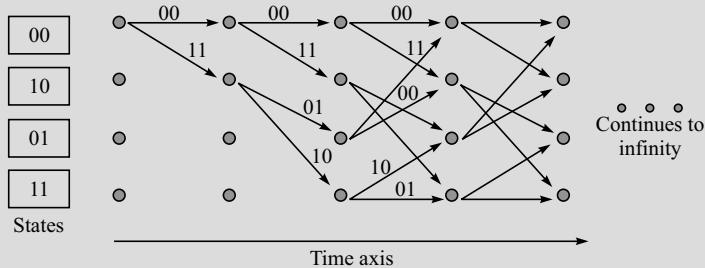


Fig. 7.9 The trellis diagram corresponding to the encoder given in Fig. 7.3.

Every node in the trellis diagram represents a state of the shift register. Since the rate of the encoder is $\frac{1}{2}$, one bit at a time is processed by the encoder. The incoming bit can be either a '0' or a '1', therefore, there are two branches emanating from each node. The top branch represents the input as '0' and the lower branch corresponds to '1'. This is true for a binary trellis diagram. In general, one would also label each branch with the input symbol to which it corresponds. Normally, the nodes that cannot be reached by starting at the top left node and moving only to the right are not shown in the trellis diagram. Corresponding to a certain state and a particular incoming bit, the encoder will produce an output. The output of the encoder is written on top of that branch. Thus, a trellis diagram gives a very easy method to encode a stream of input data. The encoding procedure using a trellis diagram is as follows. We start from the top left node (since the initial state of the encoder is [0 0]). Depending on whether a '0' or a '1' comes, we follow the upper (for '0') or the lower branch (for '1') to the next node. The encoder output is read out from the top of the branch being traversed. Again, depending on whether a '0' or a '1' comes, we follow the upper or the lower branch from the current node (state). Thus, the encoding procedure is simply following the branches on the diagram and reading out the encoder outputs that are written on top of each branch. It is like a map with the input bit stream giving the directions (e.g., Input: 0 1 1 1 0 ... \Rightarrow go up, go down, go down, go up).



Suppose we have to encode the bit stream 1 0 0 1 1 0 1 then we start from the top left node and follow the branches dictated by this input bit stream, i.e., down, up, up, down, down, up, down ..., because '0' implies taking the upper branch (up) and '1' implies taking the lower branch (down). The encoded path in the trellis diagram is shown in Fig. 7.10. The encoded sequence can be read out from the diagram as 11 01 11 11 10 10 00.... . It can be seen that there is a one-to-one correspondence between the encoded sequence and a path in the trellis diagram. Should the decoding procedure, then, just search for the most likely path in the trellis diagram? The answer is yes, as we shall see further along in this chapter!

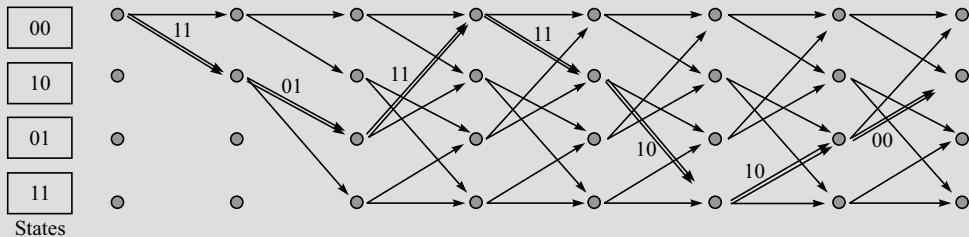


Fig. 7.10 Encoding of an input sequence using the trellis diagram.

7.3 Polynomial Description of Convolutional Codes (Analytical Representation)

LO 1

In contrast to the two pictorial representations of the convolutional codes (the state diagram and the trellis diagram), there is a very useful analytical representation of convolutional codes. The representation makes use of the delay operator, D . We have earlier seen a one-to-one correspondence between a word (vector) and a polynomial. The delay operator is used in a similar manner. For example, consider the word 10100011 with the oldest digit at the left. The analytical representation (sometimes referred to as the *transform*) of this information word $I(D)$ will be

$$I(D) = 1 + D^2 + D^6 + D^7 \quad (7.2)$$

The **Indeterminate D** is the number of time units of delay of that digit relative to the chosen time origin, which is usually taken to coincide with the first bit. In general, the sequence $i_0, i_1, i_2, i_3 \dots$ has the representation $i_0 + i_1D + i_2D^2 + i_3D^3 + \dots$

A convolutional code over $GF(q)$ with a word length $k = (m + 1)k_0$, a blocklength $n = (m + 1)n_0$ and a constraint length $v = mk_0$ can be encoded by sets of finite impulse response (FIR) filters. Each set of filters consist of k_0 FIR filters in $GF(q)$. The input to the encoder is a sequence of k_0 symbols, and the output is a sequence of n_0 symbols. Figure 7.11 shows an encoder for a binary convolutional code with $k_0 = 1$ and $n_0 = 4$. Figure 7.12 shows a convolutional filter with $k_0 = 2$ and $n_0 = 4$.

Each of the FIR filters can be represented by a polynomial of degree $\leq m$. The input stream of symbols can also be represented by a polynomial. The operation of the filter can then simply be represented as the multiplication of the two polynomials. Thus the

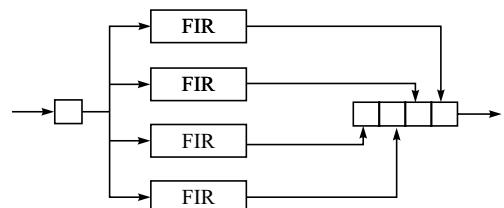


Fig. 7.11 A convolutional encoder in terms of FIR filters with $k_0 = 1$ and $n_0 = 4$.

encoder (and hence the code) can be represented by a *set of polynomials*. These polynomials are called the **Generator Polynomials** of the code. This set contains $k_0 n_0$ polynomials. The largest degree of a polynomial in this set of generator polynomials is m . We remind the reader that a block code was represented by a *single* generator polynomial. Thus we can define a **Generator Polynomial Matrix** of size $k_0 \times n_0$ for a convolutional code as follows.

$$\mathbf{G}(D) = [g_{ij}(D)] \quad (7.3)$$

$\mathbf{G}(D)$ is also called the **Transfer function**.

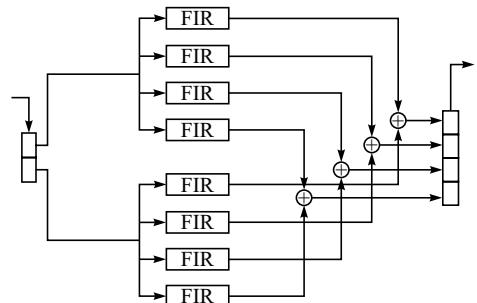


Fig. 7.12 A convolutional encoder in terms of FIR filters with $k_0 = 2$ and $n_0 = 4$. The rate of this encoder is $R = \frac{1}{2}$.

Example 7.7 Consider the convolutional encoder given in Fig. 7.13.

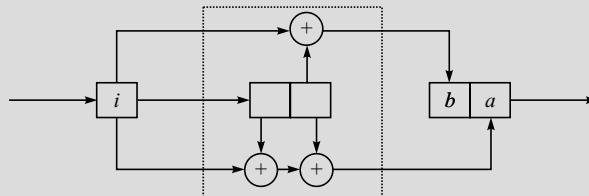


Fig. 7.13 The rate $\frac{1}{2}$ convolutional encoder with $\mathbf{G}(D) = [D^2 + D + 1 \quad D^2 + 1]$.

The first bit of the output $a = i_{n-2} + i_{n-1} + i_n$ and the second bit of the output $b = i_{n-2} + i_n$, where i_{n-l} represents the input that arrived l time units earlier. Let the input stream of symbols be represented by a polynomial. We know that multiplying any polynomial by D corresponds to a single cyclic right-shift of the elements. Therefore,

$$g_{11}(D) = D^2 + D + 1 \text{ and } g_{12}(D) = D^2 + 1$$

and the generator polynomial matrix of this encoder can be written as

$$\mathbf{G}(D) = [D^2 + D + 1 \quad D^2 + 1]$$

Next, consider the encoder circuit shown in Fig. 7.14.

In this case, $a = i_n$ and $b = i_{n-4} + i_n$. Therefore, the generator polynomial matrix of this encoder can be written as

$$\mathbf{G}(D) = [1 \quad D^4 + 1].$$

Note that the first k_0 bits ($k_0 = 1$) of the codeword frame is identical to the information frame. Hence, this is a **Systematic Convolutional Encoder**.

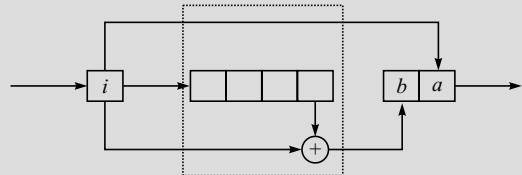


Fig. 7.14 The rate $\frac{1}{2}$ convolutional encoder with $\mathbf{G}(D) = [1 \quad D^4 + 1]$.

Example 7.8

Consider the systematic convolutional encoder represented by the following circuit (Fig. 7.15).

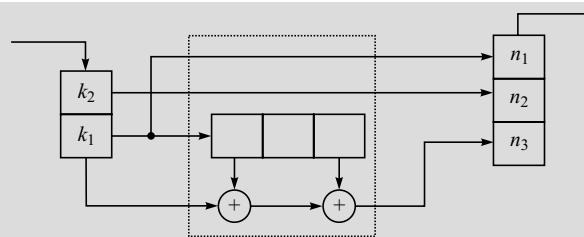


Fig. 7.15 The rate 1/3 convolutional encoder for Example 7.8.

The generator polynomial matrix of this encoder can be written as

$$\mathbf{G}(D) = \begin{bmatrix} g_{11}(D) & g_{12}(D) & g_{13}(D) \\ g_{21}(D) & g_{22}(D) & g_{23}(D) \end{bmatrix} = \begin{bmatrix} 1 & 0 & D^3 + D + 1 \\ 0 & 1 & 0 \end{bmatrix}$$



It is easy to write the generator polynomial matrix by visual inspection. The element in the i^{th} row and j^{th} column of the matrix represents the relation between the i^{th} input bit and the j^{th} output bit. To write the generator polynomial for the $(i^{\text{th}}, j^{\text{th}})$ entry of the matrix, just trace the route from the i^{th} input bit to the j^{th} output bit. If no path exists, the generator polynomial is the zero polynomial, as in the case of $g_{12}(D)$, $g_{21}(D)$ and $g_{23}(D)$. If only a direct path exists without any delay elements, the value of the generator polynomial is unity, as in $g_{11}(D)$ and $g_{22}(D)$. If the route from the i^{th} input bit to the j^{th} output bit involves a series of memory elements (delay elements), represent each delay by an additional power of D , as in $g_{13}(D)$. Note that three of the generator polynomials in the set of generator polynomials are zero. When k_0 is greater than 1, it is not unusual for some of the generator polynomials to be the zero polynomials.

We can now give the formal definitions of the word length, blocklength and the constraint length of a convolutional encoder.

Definition 7.6 Given the generator polynomial matrix $[g_{ij}(D)]$ of a convolutional code:

(i) The **Word length** of the code is

$$k = k_0 \max_{i,j} [\deg g_{ij}(D) + 1] \quad (7.4)$$

(ii) The **Blocklength** of the code is

$$n = n_0 \max_{i,j} [\deg g_{ij}(D) + 1] \quad (7.5)$$

(iii) The **Constraint length** of the code is

$$v = \sum_{i=1}^{k_0} \max_j [\deg g_{ij}(D)] \quad (7.6)$$

Recall that the input message stream $i_0, i_1, i_2, i_3 \dots$ has the polynomial representation $I(D) = i_0 + i_1 D + i_2 D^2 + i_3 D^3 + \dots + i_{k_0} D^{k_0}$ and the codeword polynomial can be written as $C(D) = c_0 + c_1 D + c_2 D^2 + c_3 D^3 + \dots + c_{n_0} D^{n_0}$. The encoding operation can simply be described as vector matrix product,

$$C(D) = I(D)\mathbf{G}(D) \quad (7.7)$$

or equivalently,

$$c_j(D) = \sum_{l=1}^{k_0} i_l(D)g_{l,j}(D) \quad (7.8)$$

Observing that the encoding operation can simply be described as vector matrix product, it can be easily shown that convolutional codes belong to the class of linear codes (exercise).

Definition 7.7 A Parity Check Matrix $\mathbf{H}(D)$ is an $(n_0 - k_0)$ by n_0 matrix of polynomials that satisfies

$$\mathbf{G}(D)\mathbf{H}(D)^T = \mathbf{0} \quad (7.9)$$

and the Syndrome Polynomial Vector, which is a $(n_0 - k_0)$ -component row vector, is given by

$$\mathbf{s}(D) = \mathbf{v}(D)\mathbf{H}(D)^T \quad (7.10)$$

where $\mathbf{v}(D)$ is the received word.

Definition 7.8 A Systematic Encoder for a convolutional code has the generator polynomial matrix of the form

$$\mathbf{G}(D) = [\mathbf{I} \mid \mathbf{P}(D)] \quad (7.11)$$

where \mathbf{I} is a k_0 by k_0 identity matrix and $\mathbf{P}(D)$ is a k_0 by $(n_0 - k_0)$ matrix of polynomials. The parity check polynomial matrix for a systematic convolutional encoder is

$$\mathbf{H}(D) = [-\mathbf{P}(D)^T \mid \mathbf{I}] \quad (7.12)$$

where \mathbf{I} is a $(n_0 - k_0)$ by $(n_0 - k_0)$ identity matrix. It follows that

$$\mathbf{G}(D)\mathbf{H}(D)^T = \mathbf{0} \quad (7.13)$$

Definition 7.9 A convolutional code whose generator polynomials $g_1(D), g_2(D), \dots, g_{n_0}(D)$ satisfy

$$\text{GCD}[g_1(D), g_2(D), \dots, g_{n_0}(D)] = D^a \quad (7.14)$$

for some a is called a **Non-catastrophic Convolutional Code**. Otherwise it is called a **Catastrophic Convolutional Code**.

Without loss of generality, one may take $a = 0$, i.e., $D^a = 1$.



Thus the task of finding a non-catastrophic convolutional code is equivalent to finding a good set of relatively prime generator polynomials. Relatively prime polynomials can be easily found by computer searches. However, what is difficult is to find a set of relatively prime generator polynomials that have good error correcting capabilities.

Example 7.9 All systematic codes are non-catastrophic because for them $g_1(D) = 1$ and therefore,

$$\text{GCD}[1, g_2(D), \dots, g_{n_0}(D)] = 1$$

Thus the systematic convolutional encoder represented by the generator polynomial matrix

$$\mathbf{G}(D) = [1 \quad D^4 + 1]$$

is non-catastrophic.

Consider the following generator polynomial matrix of a binary convolutional encoder

$$\mathbf{G}(D) = [D^2 + 1 \quad D^4 + 1]$$

We observe that $(D^2 + 1)^2 = D^4 + (D^2 + D^2) + 1 = D^4 + 1$ for binary encoder (modulo 2 arithmetic). Hence, $\text{GCD}[g_1(D), g_2(D)] = D^2 + 1 \neq 1$. Therefore, this is a catastrophic encoder.

Next consider the following generator polynomial matrix of a non-systematic binary convolutional encoder

$$\mathbf{G}(D) = [D^2 + 1 \quad D^4 + D^2 + 1]$$

The two generator polynomials are relatively prime, i.e., $\text{GCD}[g_1(D), g_2(D)] = 1$. Hence this represents a non-catastrophic convolutional encoder.

In the next section, we shall see that the distance notion of convolutional codes is an important parameter, and determines the number of errors a convolutional code can correct.

Definition 7.10 Two generator polynomial matrices (transfer functions), $\mathbf{G}_1(D)$ and $\mathbf{G}_2(D)$, are said to be **equivalent** if they generate the same convolutional code. For two equivalent convolutional encoders, $\mathbf{G}_2(D) = A(D)\mathbf{G}_1(D)$ for an invertible matrix, $A(D)$.

Example 7.10 Suppose, we have

$$\mathbf{G}_1(D) = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1+D \end{bmatrix}$$

Consider the invertible matrix,

$$A_1(D) = \begin{bmatrix} 1+D^2 & 1+D \\ 1+D & 1+D^2 \end{bmatrix}$$

Then,

$$\begin{aligned} \mathbf{G}_2(D) &= A_1(D)\mathbf{G}_1(D) \\ &= \begin{bmatrix} 1+D^2 & 1+D \\ 1+D & 1+D^2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1+D \end{bmatrix} \\ &= \begin{bmatrix} 1+D^2 & 1+D^2 & 0 \\ 1+D & 1+D & D^2 + D^3 \end{bmatrix} \end{aligned}$$

Here, $\mathbf{G}_2(D)$ and $\mathbf{G}_1(D)$ are equivalent.

Again, consider the invertible matrix,

$$A_2(D) = \begin{bmatrix} 1 & D^2 \\ D & 1 \end{bmatrix}$$

Then,

$$\begin{aligned} \mathbf{G}_3(D) &= A_2(D)\mathbf{G}_1(D) \\ &= \begin{bmatrix} 1 & D^2 \\ D & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1+D \end{bmatrix} \\ &= \begin{bmatrix} 1 & D^2 & 1+D^2 + D^3 \\ D & 1 & 1 \end{bmatrix} \end{aligned}$$

$\mathbf{G}_1(D)$, $\mathbf{G}_2(D)$ and $\mathbf{G}_3(D)$ are all equivalent, but they are not equally hardware efficient.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 1:

1. Encode the following bit stream: 1 1 1 1 0 0 0 0 ... using the encoder given in Fig. 7.5. S
Also list out the corresponding states of the memory.
2. Find the rate of the encoder (enclosed in the dotted box) shown in Fig. 7.16. S

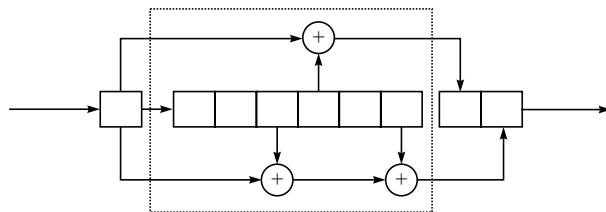


Fig. 7.16

3. Consider the encoder shown in Fig. 7.17. Find the blocklength, n . S

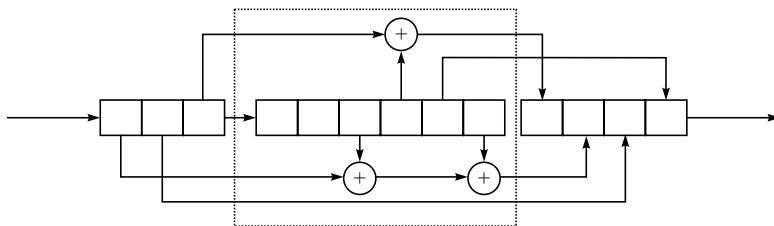


Fig. 7.17

4. Determine the Generator Polynomial Matrix for the encoder given in Fig. 7.16.
5. Is the encoder given in Example 7.3 catastrophic? S

6. Are $\mathbf{G}_1(D) = \begin{bmatrix} 1 & 0 & \frac{D}{1+D^3} \\ 0 & 1 & \frac{D^2}{1+D^3} \end{bmatrix}$ and $\mathbf{G}_2(D) = \begin{bmatrix} 1 & D^2 & D \\ D & 1 & 0 \end{bmatrix}$ equivalent? S

If you have successfully solved the above problems,
you have mastered LO 1. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/612>



Levels of Difficulty

- S **Simple:** Level 1 and Level 2 Category
- M **Medium:** Level 3 and Level 4 Category
- D **Difficult:** Level 5 and Level 6 Category

7.4 Distance Notions for Convolutional Codes

Recall that for block codes, the concept of (Hamming) distance between two codewords provided a way of quantitatively describing how different the two vectors were. It was shown that a good code must possess the maximum possible minimum distance. Convolutional codes also have a distance concept that determines how good the code is.

When a codeword from a convolutional encoder passes through a channel, errors occur from time to time. The job of the decoder is to correct these errors by processing the received vector. In principle, the convolutional codeword is infinite in length. However, the decoding decisions are made on codeword *segments* of a finite length. The number of symbols that the decoder can store is called the **Decoding Window Width**. Regardless of the size of these finite segments (decoding window width), there is always some effect of the previous frames on the current frame because of the memory of the encoder. In general, one gets a better performance by increasing the decoding window width, but one eventually reaches a point of diminishing return.

Most of the decoding procedures for decoding convolutional codes work by focussing on the errors in the first frame. If this frame can be corrected and decoded, then the first frame of information is known at the receiver end. The effect of these information symbols on the subsequent information frames can be computed and subtracted from subsequent codeword frames. Thus the problem of decoding the second codeword frame is the same as the problem of decoding the first frame. We extend this logic further. If the first f frames have been decoded successfully, the problem of decoding the $(f+1)^{\text{th}}$ frame is the same as the problem of decoding the first frame. But what happens if an in-between frame is not decoded correctly? If it is possible for a single decoding error event to induce an infinite number of *additional* errors, then the decoder is said to be subject to **Error Propagation**. In the case where the decoding algorithm is responsible for error propagation, it is termed as **Ordinary Error Propagation**. In the case where the poor choice of catastrophic generator polynomials causes error propagation, we call it **Catastrophic Error Propagation**.

Definition 7.11 The l^{th} minimum distance d_l^* of a convolutional code is equal to the smallest Hamming distance between any two initial codeword segments l frame long that are not identical in the initial frame. If $l = m + 1$, then this $(m + 1)^{\text{th}}$ minimum distance is called the **Minimum Distance** of the code and is denoted by d^* , where m is the number of information frames that can be stored in the memory of the encoder. In literature, the minimum distance is also denoted by d_{\min} .

We note here that a convolutional code is a linear code. Therefore, one of the two codewords used to determine the minimum distance of the code can be chosen to be the all zero codeword. The l^{th} minimum distance is then equal to the weight of the smallest-weight codeword segment l frames long that is non-zero in the first frame (i.e., different from the all zero frame).

Suppose the l^{th} minimum distance of a convolutional code is d_l^* . The code can correct t errors occurring in the first l frames provided

$$d_l^* \geq 2t + 1 \quad (7.15)$$

Next, put $l = m + 1$, in which case $d_l^* = d_{m+1}^* = d^*$. The code can correct t errors occurring in the first block length $n = (m + 1)n_0$ provided

$$d^* \geq 2t + 1 \quad (7.16)$$

LO 2



Explain how to perform Viterbi decoding and apply distance bounds. The performance bounds and some known good convolutional codes will also be studied.

Example 7.11 Consider the convolutional encoder of Example 7.3 (Fig. 7.5). The trellis diagram for the encoder is given in Fig. 7.18.

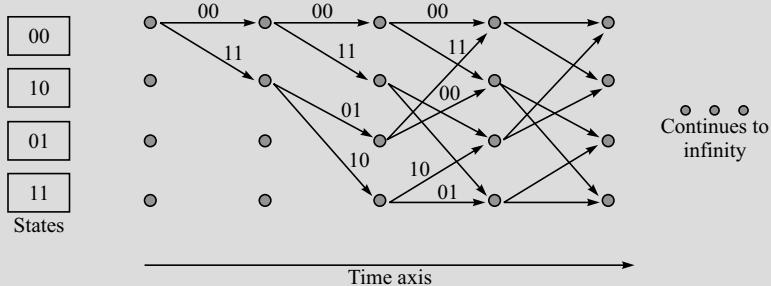


Fig. 7.18 The trellis diagram for the convolutional encoder of Example 7.3.

In this case $d_1^* = 2$, $d_2^* = 3$, $d_3^* = 5$, $d_4^* = 5$, ... We observe that $d_i^* = 5$ for $i \geq 3$. For this encoder, $m = 2$. Therefore, the minimum distance of the code is $d_3^* = d^* = 5$. This code can correct $(d^* - 1)/2 = 2$ random errors that occur in one blocklength, $n = (m + 1)n_0 = 6$.

Definition 7.12 The **free distance** of a convolutional code is given by

$$d_{\text{free}} = \max_l [d_l] \quad (7.17)$$

It follows that $d_{m+1} \leq d_{m+2} \leq \dots \leq d_{\text{free}}$.

 The term d_{free} was first coined by Massey in 1969 to denote a type of distance that was found to be an important parameter for the decoding techniques of convolutional codes. Since, d_{free} represents the minimum distance between arbitrarily long (possibly infinite) encoded sequences, d_{free} is also denoted by d_∞ in literature. The parameter d_{free} can be directly calculated from the trellis diagram. The free distance d_{free} is the minimum weight of a path that deviates from the all zero path and later merges back into the all zero path at some point further down the trellis as depicted in Fig. 7.19. Searching for a code with large minimum distance and large free distance is a tedious process, and is often done using a computer. Clever techniques have been designed that reduce the effort by avoiding exhaustive searches. Most of the good convolutional codes known today have been discovered by computer searches.

Definition 7.13 The **free length** n_{free} of a convolutional code is the length of the non-zero segment of a smallest weight convolutional codeword of non-zero weight. Thus, $d_l = d_{\text{free}}$ if $l = n_{\text{free}}$, and $d_l < d_{\text{free}}$ if $l < n_{\text{free}}$. In literature, n_{free} is also denoted by n_∞ .

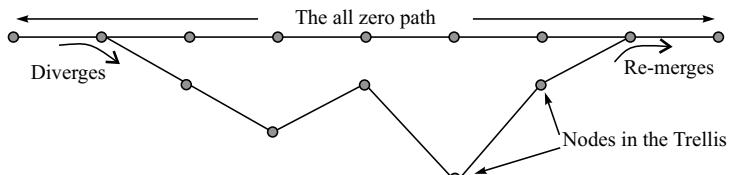


Fig. 7.19 The free distance d_{free} is the minimum weight of the path that diverges from the all zero path and later merges back.

Example 7.12 Consider the convolutional encoder of Example 7.3 (Fig. 7.5). For this encoder, $d_{\text{free}} = 5$. There are usually more than one pair of paths that can be used to calculate d_{free} . The two paths that have been used to calculate d_{free} are represented in Fig. 7.20 by double lines. In this example, $d_{\text{min}} = d_{\text{free}}$.

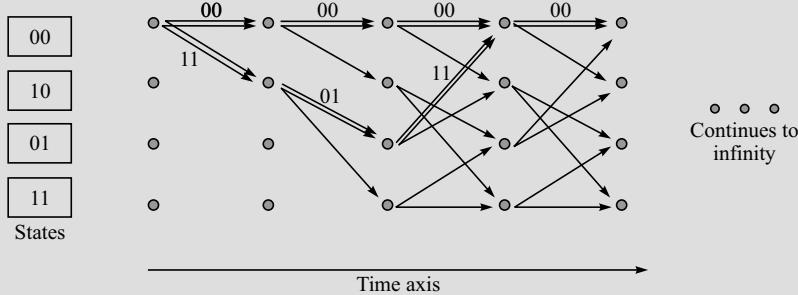


Fig. 7.20 The pair of paths that is used to calculate d_{free} in the trellis diagram.

The free length of the convolutional code is $n_{\text{free}} = 6$. In this example, the n_{free} is equal to the blocklength n of the code. In general it can be longer than the blocklength.

7.5 The Generating Function

LO 2

The performance of a convolutional code depends on its free distance, d_{free} . Since convolutional codes are a sub class of linear codes, the set of Hamming distances between coded sequences is the same as the set of distances of the coded sequences from the all zero sequence. Therefore, we can consider the distance structure of the convolutional codes with respect to the all zero sequence without loss in generality. In this section we shall study an elegant method of determining the free distance, d_{free} , of a convolutional code.

To find d_{free} we need the set of paths that diverge from the all zero path and later remerge. The brute force (and time consuming, not to mention, exasperating) method would be to determine the instances of all possible paths from the trellis diagram. Another way to find out the d_{free} of a convolutional code is to use the concept of a **generating function**, whose expansion provides all the distance information directly. The generating function can be understood from Example 7.13.

Example 7.13 Consider again the convolutional encoder of Example 7.3 (Fig. 7.5). The state diagram of the encoder is given in Fig. 7.7. We now construct a modified state diagram as shown in Fig. 7.21.

The branches of this modified state diagram are labelled by branch gain D^i , $i = 0, 1, 2$, where the exponent of D denotes the Hamming weight of the branch. Note that the self-loop S_0 has been neglected as it does not contribute to the distance property of the code. Circulating around this loop simply generates the all zero sequence. Note that S_0 has been split into two states, initial and final. Any path that diverges from and later merges back

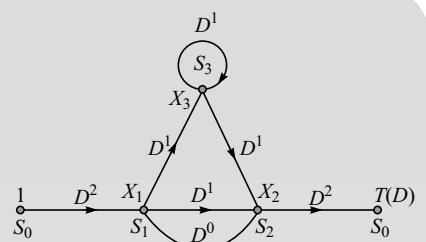


Fig. 7.21 The modified state diagram of the convolutional encoder shown in Fig. 7.5.

to state S_0 can be thought of equivalently traversing along the branches of this modified state diagram, starting from the initial S_0 and ending at the final S_0 . Hence this modified state diagram encompasses all possible paths that diverge from and then later merge into the all zero path in the trellis diagram.

We can find the distance profile of the convolutional code using the state equations of this modified state diagram. These state equations are

$$\begin{aligned} X_1 &= D^2 + X_2 \\ X_2 &= DX_1 + DX_3 \\ X_3 &= DX_1 + DX_3 \\ T(D) &= D^2X_2 \end{aligned} \quad (7.18)$$

where X_i are dummy variables. Upon solving these equations simultaneously we obtain the generating function

$$\begin{aligned} T(D) &= \frac{D^5}{1-2D} \\ &= D^5 + 2D^6 + 4D^7 + \dots + 2^k D^{k+5} + \dots \\ &= \sum_{d=5}^{\infty} a_d D^d. \end{aligned} \quad (7.19)$$

Note that the expression for $T(D)$ can also be (easily) obtained by the Mason's gain formula, which is well known to the students of digital signal processing.

Following conclusions can be drawn from the series expansion of the generating function:



- (i) There are an infinite number of possible paths that diverge from the all zero path and later remerge (this is also intuitive).
 - (ii) There is only one path with Hamming distance 5, two paths with Hamming distance 6, and in general 2^k paths with Hamming distance $k+5$ from the all zero path.
 - (iii) The free Hamming distance d_{free} for this code is 5. There is only one path corresponding to d_{free} .
- Example 7.12 explicitly illustrates the pair of paths that result in $d_{\text{free}} = 5$.

We now introduce two new labels in the modified state diagram. To enumerate the length of a given path, the label L is attached to each branch. So every time we traverse along a branch we experience an increment in the path length counter. We also add the label I^i , to each branch to enumerate the Hamming weight of the input bits associated with each branch of the modified state diagram (see Fig. 7.22).

The state equations in this case would be

$$\begin{aligned} X_1 &= D^2LI + LIX_2 \\ X_2 &= DLX_1 + DLX_3 \\ X_3 &= DLIX_1 + DLIX_3 \end{aligned}$$

and the **Augmented Generating Function** is

$$T(D) = D^2LX_2 \quad (7.20)$$

On solving these simultaneous equations we obtain

$$\begin{aligned} T(D, L, I) &= \frac{D^5 L^3 I}{1 - DL(L+1)I} \\ &= D^5 L^3 I + D^6 L^4 (L+1)I^2 + \dots + D^{k+5} L^{3+k} (L+1)^k I^{k+1} + \dots \end{aligned} \quad (7.21)$$

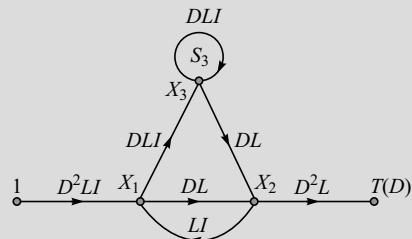


Fig. 7.22 The modified state diagram used to determine the augmented generating function.

Further conclusions from the series expansion of the augmented generating function are as follows:

- (i) The path with the minimum Hamming distance of 5 has length equal to 3.
- (ii) The input sequence corresponding to this path has weight equal to 1.
- (iii) There are two paths with Hamming distance equal to 6 from the all zero path. Of these, one has a path length of 4 and the other a path length of 5 (observe the power of L in the second term in the summation). Both these paths have an input sequence weight of 2.



Recall that the free distance, d_{free} , is the minimum weight of a path that deviates from the all zero path and later merges back into the all zero path at some point further down the trellis. So, in general, we can write

$$T(D) = \sum_{d=d_{\text{free}}}^{\infty} a_d D^d \quad (7.22)$$

where, $a_{d_{\text{free}}}$ denotes the number of paths with weight d_{free} . Intuitively, if we have to choose between two codes with the same d_{free} , we must pick the one with the smaller $a_{d_{\text{free}}}$, since there would be fewer numbers of paths with smallest weight.

In the next section, we will study the matrix description of convolutional codes. The matrix representation of convolutional codes, as we shall shortly discover, is a bit more complicated than the matrix description of linear block codes.

7.6 Matrix Description of Convolutional Codes

LO 2

A convolutional code can be described by an infinite number of infinitely long codewords (visualise the trellis diagram). We have also seen that convolutional codes belong to the class of linear codes. Therefore, they can be described by an infinite generator matrix. As expected, the matrix description of convolutional codes is messier than that of the linear block codes.

Let the generator polynomials of a convolutional code be represented by

$$g_{ij}(D) = \sum_l g_{ijl} D^l \quad (7.23)$$

In order to obtain a generator matrix, the g_{ijl} coefficients are arranged in a matrix format. For each l , let \mathbf{G}_l be a k_0 by n_0 matrix.

$$\mathbf{G}_l = [g_{ijl}] \quad (7.24)$$

Then the generator matrix for the convolutional code that has been truncated to a block code of block-length n is

$$\mathbf{G}^{(n)} = \begin{bmatrix} \mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_2 & \dots & \mathbf{G}_m \\ \mathbf{0} & \mathbf{G}_0 & \mathbf{G}_0 & \dots & \mathbf{G}_{m-1} \\ \mathbf{0} & \mathbf{0} & \mathbf{G}_0 & \dots & \mathbf{G}_{m-2} \\ \vdots & \vdots & \vdots & & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{G}_0 \end{bmatrix} \quad (7.25)$$

where $\mathbf{0}$ is a k_0 by n_0 matrix of zeroes and m is the length of the shift register used to generate the code. The generator matrix for the convolutional code is given by

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_2 & \dots & \mathbf{G}_m & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{G}_0 & \mathbf{G}_1 & \dots & \mathbf{G}_{m-1} & \mathbf{G}_m & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{0} & \mathbf{G}_0 & \dots & \mathbf{G}_{m-2} & \mathbf{G}_{m-1} & \mathbf{G}_m & \mathbf{0} & \dots \\ \vdots & \vdots & & & & & & & \vdots \end{bmatrix} \quad (7.26)$$

The matrix extends infinitely far down and to the right. For a systematic convolutional code, the generator matrix can be written as

$$\mathbf{G} = \begin{bmatrix} \mathbf{I} & \mathbf{P}_0 & \mathbf{0} & \mathbf{P}_1 & \mathbf{0} & \mathbf{P}_2 & \dots & \mathbf{0} & \mathbf{P}_m & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{P}_0 & \mathbf{0} & \mathbf{P}_1 & \dots & \mathbf{0} & \mathbf{P}_{m-1} & \mathbf{0} & \mathbf{P}_m & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{P}_0 & \dots & \mathbf{0} & \mathbf{P}_{m-2} & \mathbf{0} & \mathbf{P}_{m-1} & \mathbf{0} & \mathbf{P}_m & \dots \\ \vdots & \vdots \end{bmatrix} \quad (7.27)$$

where \mathbf{I} is a k_0 by k_0 identity matrix, $\mathbf{0}$ is a k_0 by k_0 matrix of zeroes and $\mathbf{P}_0, \mathbf{P}_1, \dots, \mathbf{P}_m$ are k_0 by $(n_0 - k_0)$ matrices. The parity check matrix can then be written as

$$\mathbf{H} = \begin{bmatrix} \mathbf{P}_0^T & -\mathbf{I} & & & & & \dots \\ \mathbf{P}_1^T & \mathbf{0} & \mathbf{P}_0^T & -\mathbf{I} & & & \\ \mathbf{P}_2^T & \mathbf{0} & \mathbf{P}_1^T & \mathbf{0} & \mathbf{P}_0^T & -\mathbf{I} & \\ \vdots & & \vdots & & \vdots & & \vdots \\ \mathbf{P}_m^T & \mathbf{0} & \mathbf{P}_{m-1}^T & \mathbf{0} & \mathbf{P}_{m-2}^T & \mathbf{0} & \dots & \mathbf{P}_0^T & -\mathbf{I} & \dots \\ & & \mathbf{P}_m^T & \mathbf{0} & \mathbf{P}_{m-1}^T & \mathbf{0} & \dots & & & \\ & & & & \mathbf{P}_m^T & \mathbf{0} & \dots & & & \dots \end{bmatrix} \quad (7.28)$$

Example 7.14 Consider the convolutional encoder shown in Fig. 7.23. Let us first write the generator polynomial matrix for this encoder. To do so, we just follow individual inputs to the outputs, one by one, and count the delays. The generator polynomial matrix is obtained as

$$G(D) = \begin{bmatrix} D + D^2 & D^2 & D + D^2 \\ D^2 & D & D \end{bmatrix}$$

The generator polynomials are $g_{11}(D) = D + D^2$, $g_{12}(D) = D^2$, $g_{13}(D) = D + D^2$, $g_{21}(D) = D^2$, $g_{22}(D) = D$ and $g_{23}(D) = D$.

To write out the matrix \mathbf{G}_0 , we look at the constants (coefficients of D^0) in the generator polynomials. Since there are no constant terms in any of the generator polynomials,

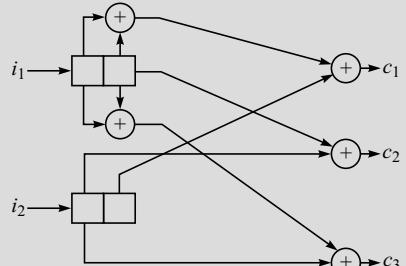


Fig. 7.23 A rate 2/3 convolutional encoder.

$$\mathbf{G}_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Next, to write out the matrix \mathbf{G}_1 , we look at the coefficients of D^1 in the generator polynomials. The 1st row, 1st column entry of the matrix \mathbf{G}_1 corresponds to the coefficients of D^1 in $g_{11}(D)$. The 1st row, 2nd column entry corresponds to the coefficients of D^1 in $g_{12}(D)$, and so on. Thus,

$$\mathbf{G}_1 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Similarly, we can write

$$\mathbf{G}_2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

The generator matrix can now be written as

$$\mathbf{G} = \left[\begin{array}{ccc|ccc|ccc|ccc|c} 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ \hline & & & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & \dots \\ & & & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ \hline & : & & : & : & : & : & : & : & 1 & 0 & 1 & \dots \\ & : & & : & : & : & : & : & : & 0 & 1 & 1 & \dots \end{array} \right]$$

Our next task is to look at an efficient decoding strategy for the convolutional codes. One of the very popular decoding methods, the Viterbi decoding technique, will be discussed in detail.

7.7 Viterbi Decoding of Convolutional Codes

LO 2

There are three important decoding techniques for convolutional codes: threshold decoding, sequential decoding and the Viterbi decoding. The **Sequential Decoding** technique was proposed by Wozencraft in 1957. Sequential decoding has the advantage that it can perform very well with long-constraint-length convolutional codes, but it has a variable decoding time. **Threshold Decoding**, also known as **Majority Logic Decoding**, was proposed by Massey in 1963 as a part of his doctoral thesis at MIT. Threshold decoders were the first commercially produced decoders for convolutional codes. **Viterbi decoding** was developed by Andrew J. Viterbi in 1967. In the late 1970s, Viterbi decoding became the dominant technique for convolutional codes. Viterbi decoding had the advantages of (i) a highly satisfactory bit error performance, (ii) high speed of operation, (iii) ease of implementation and (iv) low cost. Threshold decoding lost its popularity especially because of its inferior bit error performance. Threshold decoding is conceptually and practically closest to block decoding. It requires the calculation of a set of syndromes, just as in the case of block codes. In this case, the syndrome is a sequence because the information and check digits occur as sequences.



INDUSTRIAL RELEVANCE

Viterbi decoding has the advantage that it has a fixed decoding time. It is well-suited to hardware decoder implementation. But its computational requirements grow exponentially as a function of the constraint length, so it is usually limited in practice to constraint lengths of $v = 9$ or less. As of mid-2000, some leading companies have claimed to produce a $v = 15$ Viterbi decoder that operates at rates up to 2 Mbps.

Since the time when Viterbi proposed his algorithm, other researchers have expanded on his work by finding good convolutional codes, exploring the performance limits of the technique and varying decoder design parameters to optimise the implementation of the technique in hardware and software. The Viterbi decoding algorithm is also used in decoding Trellis Coded Modulation (TCM), the technique used in telephone-line modems to squeeze high ratios of bits-per-second to Hertz out of 3 kHz-bandwidth analogue telephone lines. We shall see more of TCM in the next chapter. For years, convolutional coding with Viterbi decoding has been the predominant FEC (forward error correction) technique used in space communications, particularly in geostationary satellite communication networks, such as VSAT (very small aperture terminal) networks. The most common variant used in VSAT networks is rate 1/2 convolutional coding using a code with a constraint length $v = 7$. With this code, one can transmit binary or quaternary phase-shift-keyed (BPSK or QPSK) signals with at least 5 dB less power than without coding. That's a reduction in Watts of more than a factor of three! This is very useful in reducing transmitter and antenna cost or permitting increased data rates given the same transmitter power and antenna sizes. According to one estimate, in 2006, Viterbi decoders were used in about one billion cellular phones, which is probably the largest number in any application.

We will now consider how to decode convolutional codes using the Viterbi decoding algorithm. The nomenclature used here is that we have a message vector \mathbf{i} from which the encoder generates a code vector \mathbf{c} that is sent across a discrete memoryless channel. The received vector \mathbf{r} may differ from the transmitted vector \mathbf{c} (unless the channel is ideal or we are very lucky!). The decoder is required to make an estimate of the message vector. Since there is a one-to-one correspondence between code vector and message vector, the decoder makes an estimate of the code vector.

Optimum decoding will result in a minimum probability of decoding error. Let $p(\mathbf{r}|\mathbf{c})$ be the conditional probability of receiving \mathbf{r} given that \mathbf{c} was sent. We can state that the optimum decoder is the maximum likelihood decoder with a decision rule to choose the code vector estimate $\hat{\mathbf{c}}$ for which the log-likelihood function $\ln p(\mathbf{r}|\mathbf{c})$ is maximum.

If we consider a binary symmetric channel where the vector elements of \mathbf{c} and \mathbf{r} are denoted by c_i and r_i then we have

$$p(\mathbf{r} | \mathbf{c}) = \prod_{i=1}^N p(r_i | c_i) \quad (7.29)$$

where N is the length of the sequence. Hence the log-likelihood function equals

$$\ln p(\mathbf{r} | \mathbf{c}) = \sum_{i=1}^N \ln p(r_i | c_i) \quad (7.30)$$

Let us assume

$$p(r_i | c_i) = \begin{cases} p, & r_i \neq c_i \\ 1-p, & r_i = c_i \end{cases} \quad (7.31)$$

If we suppose that the received vector differs from the transmitted vector in exactly d positions (the Hamming distance between vectors \mathbf{c} and \mathbf{r}), we may rewrite the log-likelihood function as

$$\begin{aligned}\ln p(\mathbf{r} | \mathbf{c}) &= d \ln p + (N - d) \ln(1 - p) \\ &= d \ln \left(\frac{p}{1 - p} \right) + N \ln(1 - p)\end{aligned}\quad (7.32)$$

We can assume the probability of error $p < \frac{1}{2}$ and we note that $N \ln(1 - p)$ is a constant for all code vectors. Now we can make the statement that *the maximum likelihood decoding rule for a binary symmetric channel is to choose the code vector estimate $\hat{\mathbf{c}}$ that minimises the Hamming distance between the received vector \mathbf{r} and the transmitted vector \mathbf{c} .*

For soft decision decoding in additive white Gaussian noise (AWGN) channel with single-sided noise power N_0 , the likelihood function is given by

$$\begin{aligned}p(\mathbf{r} | \mathbf{c}) &= \prod_{i=1}^N \frac{1}{\sqrt{\pi N_0}} e^{-\frac{|r_i - c_i|^2}{N_0}} \\ &= \left(\frac{1}{\pi N_0} \right)^{\frac{1}{2}} \exp \left(-\frac{1}{N_0} \sum_{i=1}^N |r_i - c_i|^2 \right)\end{aligned}\quad (7.33)$$

 **Intuition** Thus the maximum likelihood decoding rule for the AWGN channel with soft decision decoding is to *minimise the squared Euclidean distance* between \mathbf{r} and \mathbf{c} . This squared Euclidean distance is given by

$$d_E^2(\mathbf{r} | \mathbf{c}) = \sum_{i=1}^N |r_i - c_i|^2 \quad (7.34)$$

Viterbi decoding works by choosing that trial information sequence, the encoded version of which is *closest* to the received sequence. Here, Hamming distance will be used as a measure of closeness between two sequences. The Viterbi decoding procedure can be easily understood by the following example.

Example 7.15 Consider the rate 1/3 convolutional encoder given in Fig. 7.24 and the corresponding trellis diagram.

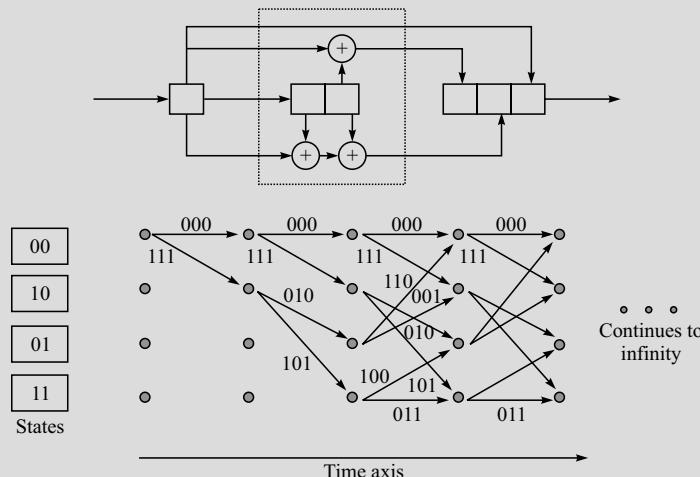


Fig. 7.24 A rate 1/3 convolutional encoder and its trellis diagram.

Suppose the transmitted sequence was the all zero sequence. Let the received sequence be

$$\mathbf{r} = 010000100001 \dots$$

Since it is a $1/3$ rate encoder, we first segment the received sequence in groups of three bits (because $n_0 = 3$), i.e.,

$$\mathbf{r} = 010\ 000\ 100\ 001 \dots$$

The task at hand is to find out the most likely path through the trellis. Since a path must pass through nodes in the trellis, we will try to find out which nodes in the trellis belong to the most likely path. At any time, every node has two incoming branches (\rightarrow). We simply determine which of these two branches belongs to a more likely path (and discard the other). We take this decision based on some metric (Hamming distance). In this way we just have to retain only one path per node and the metric of that path. In this example, we will have to retain only four paths as we progress with our decoding (since we have only 4 states in our trellis).

Let us consider the first branch of the trellis which is labelled 000. We find the Hamming distance between this branch and the first received frame length, 010. The Hamming distance $d(000, 010) = 1$. Thus the metric for this first branch is 1, and is called the **branch metric**. Upon reaching the top node from the starting node, this branch has accumulated a metric = 1. Next we compare the received frame length with the lower branch, which terminates

at the second node from the top. The Hamming distance in this case is $d(111, 010) = 2$. Thus the metric for this first branch is 2. At each node we write the total metric accumulated by the path, called the **path metric**. The path metrics are marked by circled numbers in the trellis diagram in Fig. 7.25. At the subsequent stages of decoding when two paths terminate at every node, we will retain the path with the smaller value of the metric.

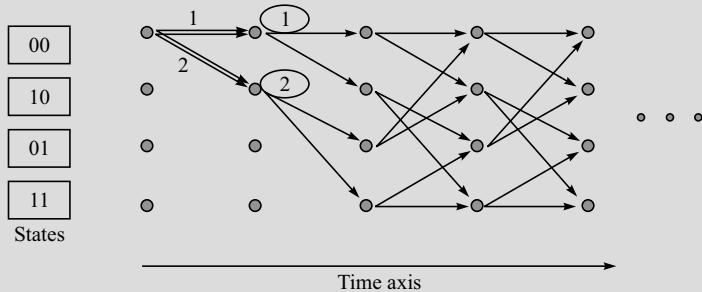


Fig. 7.25 The path metric after the 1st step of Viterbi decoding.

We now move to the next stage of the trellis. The Hamming distance between the branches are computed with respect to the second frame received, 000. The branch metrics for the two branches emanating from the topmost node are 0 and 3. The branch metrics for the two branches emanating from the second node are 1 and 2. The total path metric is marked by circled numbers in the trellis diagram shown in Fig. 7.26.

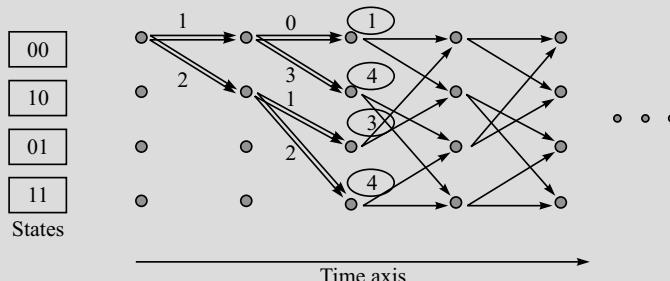


Fig. 7.26 The path metric after the 2nd step of Viterbi decoding.

We now proceed to the next stage. We again compute the branch metrics and add them to the respective path metrics to get the new path metrics. Consider the top-most node at this stage. Two branches terminate at this node. The path coming from node 1 of the previous stage has a path metric of 2 and the path coming from node 1 of the previous stage has a path metric of 4. The path with a lower metric is retained and the other path is discarded. The trellis diagram shown in Fig. 7.27 gives the *surviving paths* (double lines) and the path metrics (circled numbers). Viterbi called these surviving paths as **Survivors**.

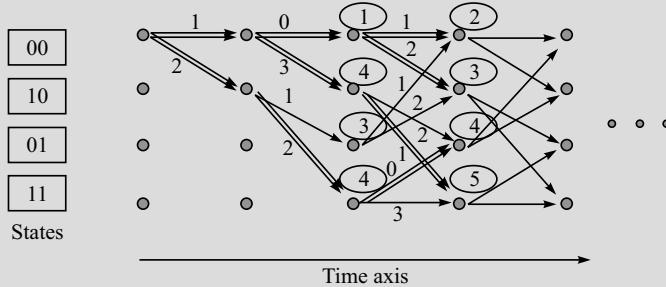


Fig. 7.27 The path metric after the 3rd step of Viterbi decoding.

We continue this procedure for Viterbi decoding for the next stage. The final branch metrics and path metrics are shown in Fig. 7.28. At the end we pick the path with the minimum metric, which corresponds to the all zero path in this example. Thus the decoding procedure has been able to correctly decode the received vector. We note that node 2 receives two paths with equal path metrics. We would have arbitrarily chosen one of them as the surviving path (by tossing a fair coin!) if we were to proceed further with the decoding process.

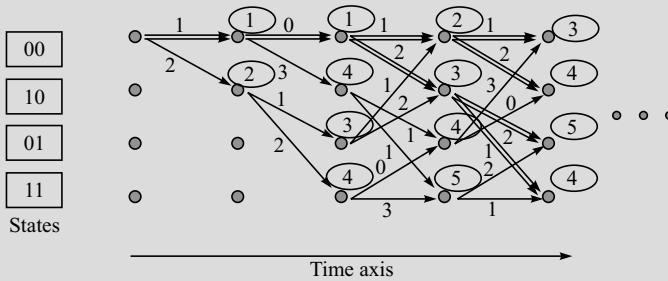


Fig. 7.28 The path metric after the 4th step of Viterbi decoding.

The minimum distance for this code is $d^* = 7$. The number of errors that it can correct per frame length is equal to

$$t = \lfloor (d^* - 1)/2 \rfloor = \lfloor (6 - 1)/2 \rfloor = 2$$

In this example, the maximum number of errors per frame length was 1.

Consider the set of surviving paths at the r^{th} frame time. If all the surviving paths cross through the same nodes then a decision regarding the most likely path transmitted can be made up to the point where the nodes are common. To build a practical Viterbi decoder, one must choose a decoding window width w , which is usually several times as big as the block length. At a given frame time, f , the decoder examines all the surviving paths to see if they agree in the first branch. This branch defines a decoded

information frame and is passed out of the decoder. In the previous example of Viterbi decoding, we see that by the time the decoder reaches the 4th frame, all the surviving paths agree in their first decoded branch (called a well-defined decision). The decoder drops the first branch (after delivering the decoded frame) and takes in a new frame of the received word for the next iteration. If again, all the surviving paths pass through the same node of the oldest surviving frame, then this information frame is decoded. The process continues in this way indefinitely.



Interpretation

If the decoding window w is chosen long enough, then a well-defined decision can be reached almost always. A well-designed code will lead to correct decoding with a high probability.

Note that a well-designed code carries meaning only in the context of a particular channel.

The random errors induced by the channel should be within the error correcting capability of the code. The Viterbi decoder can be visualised as a sliding window through which the trellis is viewed (see Fig. 7.29). The window slides to the right as new frames are processed. The surviving paths are marked on the portion of the trellis which is visible through the window. As the window slides, new nodes appear on the right and some of the surviving paths are extended to these new nodes while the other paths disappear.

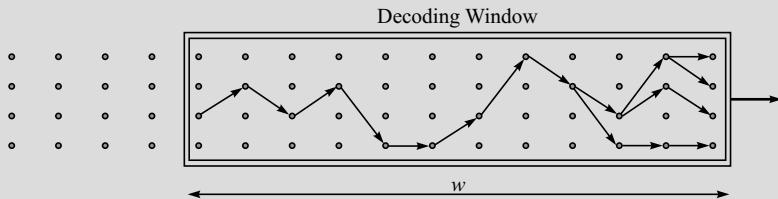


Fig. 7.29 The Viterbi decoder as a sliding window through which the trellis is viewed.



If the surviving paths do not go through the same node, we label it a **decoding failure**. The decoder can break the deadlock using any arbitrary rule. To this limited extent the decoder becomes an incomplete decoder. Let us revert to the previous example. At the end of the 4th stage, the surviving paths could as well be chosen as shown in Fig. 7.30, which will render the decoder as incomplete.

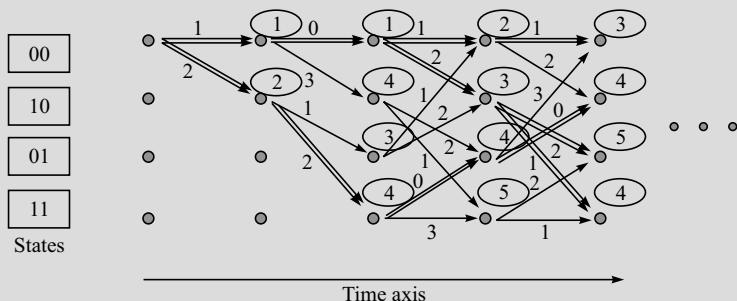


Fig. 7.30 Example of an incomplete decoder in Viterbi Decoding process.

It may be possible that in some cases the decoder reaches a well-defined decision, but a wrong one! If this happens, the decoder has no way of knowing that it has taken a wrong decision. Based on this wrong decision, the decoder will take more wrong decisions. However, if the code is non-catastrophic, the decoder will recover from the errors.

Any t error correcting (n, k) convolutional code will also correct a burst error of length t (yes, it is obvious!). Longer bursts can be corrected by interleaving the codewords. Suppose, we take j copies of the original encoder capable of correcting t errors and merge the codewords by alternating the symbols. In this process, we effectively obtain a (jn, jk) convolutional code that can correct jt errors. This technique of interleaving creates one convolutional code from another convolutional code. If the generator polynomial matrix of the original code is given by $\mathbf{G}(D)$, then the generator polynomial matrix for the interleaved code is given by $\mathbf{G}(D^j)$.

Example 7.16 Consider the rate 1/2, (14, 7) convolutional encoder given by the generator polynomial matrix $\mathbf{G}(D) = [1 \ D^6 + D^5 + D^2 + 1]$. This can correct $t = 2$ errors. If we take five copies of this encoder and interleave the codewords, we obtain a (70, 35) code capable of correcting bursts of length 10. Note that we did not change the code rate. Here, we have interleaved the outputs of $j = 5$ encoders. Hence, the generator polynomial matrix decodes

$$\mathbf{G}_5(D) = \mathbf{G}(D^j) = [1 \ D^{30} + D^{25} + D^{10} + 1]$$

The next section deals with some distance bounds for convolutional codes. These bounds will help us to compare different convolutional coding schemes.

7.8 Distance Bounds for Convolutional Codes

LO 2

Upper bounds can be computed on the minimum distance of a convolutional code that has a rate $R = \frac{k_0}{n_0}$ and a constraint length $v = mk_0$. These bounds are similar in nature and derivation to those for block codes, with blocklength corresponding to constraint length. However, as we shall see the bounds are not very tight. These bounds just give us a rough idea of how good the code is. Here we present the bounds (without proof) for binary codes.

For rate R and constraint length, let d be the largest integer that satisfies

$$H\left(\frac{d}{n_0 v}\right) \leq 1 - R \quad (7.35)$$

Then at least one binary convolutional code exists with minimum distance d for which the above inequality holds. Here $H(x)$ is the familiar entropy function for a binary alphabet

$$H(x) = -x \log_2 x - (1-x) \log_2 (1-x), \quad 0 \leq x \leq 1$$

For a binary code with $R = 1/n_0$ the minimum distance d_{\min} satisfies

$$d_{\min} \leq \lfloor (n_0 v + n_0)/2 \rfloor \quad (7.36)$$

where $\lfloor I \rfloor$ denotes the largest integer less than or equal to I .

An upper bound on d_{free} is given by (Heller, 1968)

$$d_{\text{free}} = \min_{j \geq 1} \left\lceil \frac{n_0}{2} \frac{2^j}{2^j - 1} (v + j - 1) \right\rceil \quad (7.37)$$

To calculate the upper bound, the right hand side should be plotted for different integer values of j . The upper bound is the minimum of this plot.

Example 7.17 Let us apply the distance bounds on the convolutional encoder given in Example 7.3. We will first apply the bound given by (7.35). For this encoder, $k_0 = 1$, $n_0 = 2$, $R = 1/2$ and $v = 2$.

$$H\left(\frac{d}{n_0 v}\right) = H(d/4) \leq 1 - R = 1/2 \Rightarrow H(d/4) \leq 0.5$$

But we have,

$$\begin{aligned} H(0.11) &= -0.11\log_2 0.11 - (1 - 0.11)\log_2(1 - 0.11) = 0.4999, \text{ and} \\ H(0.12) &= -0.12\log_2 0.12 - (1 - 0.12)\log_2(1 - 0.12) = 0.5294 \end{aligned}$$

Therefore, $d/4 \leq 0.11$, or $d \leq 0.44$. The largest integer d that satisfies this bound is $d = 0$. This implies that at least one binary convolutional code exists with minimum distance $d = 0$. This statement does not say much (i.e., the bound is not strict enough for this encoder)!

Next consider the encoder shown in Fig. 7.31.

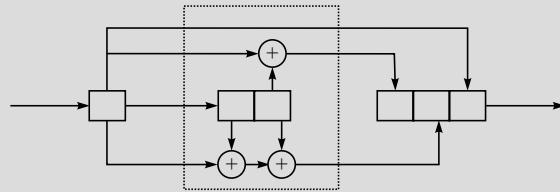


Fig. 7.31 Convolutional encoder for Example 7.17.

For this encoder,

$$\mathbf{G}(D) = [1 \quad 1 + D + D^2 \quad 1 + D^2]$$

$k_0 = 1$, $n_0 = 3$, $R = 1/3$ and $v = 2$.

$$H\left(\frac{d}{n_0 v}\right) = H(d/6) \leq 1 - R = 2/3 \Rightarrow H(d/6) \leq 0.6667$$

But we have,

$$\begin{aligned} H(0.17) &= -0.17\log_2 0.17 - (1 - 0.17)\log_2(1 - 0.17) = 0.6577, \text{ and} \\ H(0.18) &= -0.18\log_2 0.18 - (1 - 0.18)\log_2(1 - 0.18) = 0.6801 \end{aligned}$$

Therefore, $d/6 \leq 0.17$, or $d \leq 1.02$. The largest integer d that satisfies this bound is $d = 1$. Then at least one binary convolutional code exists with minimum distance $d = 1$. This is a very loose bound.

Let us determine the Heller bound on d_{free} , as given by (7.37). The plot of the function

$$d(j) = \lfloor (n_0/2)(2^j/(2^j - 1))(v + j - 1) \rfloor$$

for different integer values of j is given in Fig. 7.32.

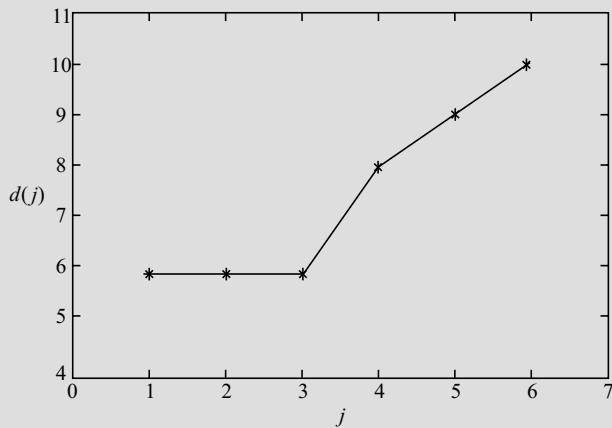


Fig. 7.32 The Heller bound plot.

From Fig. 7.32 we see that the upper bound on the free distance of the code is $d_{\text{free}} \leq 7$. The actual value of $d_{\text{free}} = 6$, as can be seen from the trellis diagram for the encoder (Fig. 7.33). The two paths that have been used to calculate d_{\min} are shown in Fig. 7.33 by double lines. In this example, $d_{\min} = d_{\text{free}} = 6$.

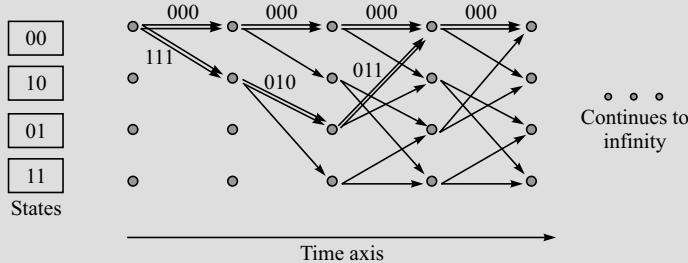


Fig. 7.33 The trellis diagram for the convolutional encoder given in Fig. 7.31.

7.9 Performance Bounds

LO 2

One of the useful performance criteria for convolutional codes is the bit error probability P_b . The bit error probability or the bit error rate (a misnomer!) is defined as the expected number of decoded information bit errors per information bit. Instead of obtaining an exact expression for P_b , typically, an upper bound on the error probability is calculated. We will first determine the **first event error probability**, which is the probability of error for sequences that merge with the all zero (correct) path for the first time at a given node in the trellis diagram.

Since convolutional codes are linear, let us assume that the all zero codeword is transmitted. An error will be made by the decoder if it chooses the incorrect path c' instead of the all zero path. Let c' differ from the

all zero path in d bits. Therefore, a wrong decision will be made by the maximum likely decoder if more than $\left\lfloor \frac{d}{2} \right\rfloor$ errors occur, where $\lfloor x \rfloor$ is the largest integer less than or equal to x . If the channel transition probability is p , then the probability of error can be upper bounded as follows.

$$P_d \leq \left[2\sqrt{p(1-p)} \right]^d \quad (7.38)$$

Now, there would be many paths with different distances that merge with the correct path at a given time for the first time. The upper bound on the first error probability can be obtained by summing the error probabilities of all such possible paths:

$$P_e \leq \sum_{d=d_{free}}^{\infty} a_d P_d \quad (7.39)$$

where, a_d is the number of codewords of Hamming distance d from the all zero codeword. Comparing (7.22) and (7.39) we obtain

$$P_e \leq T(D) \Big|_{D=2\sqrt{p(1-p)}} \quad (7.40)$$

The bit error probability, P_b , can now be determined as follows. P_b can be upper bounded by weighting each pairwise error probability, P_d , in (7.38) by the number of incorrectly decoded information bits for the corresponding incorrect path n_d . For a rake k/n encoder, the average P_b is

$$P_b \leq \frac{1}{k} \sum_{d=d_{free}}^{\infty} a_d n_d P_d \quad (7.41)$$

It can be shown that

$$\frac{\partial T(D, I)}{\partial I} \Big|_{I=1} = \sum_{d=d_{free}}^{\infty} a_d n_d D^d \quad (7.42)$$

Thus,

$$P_b \leq \frac{1}{k} \frac{\partial T(D, I)}{\partial I} \Big|_{I=1, D=2\sqrt{p(1-p)}} \quad (7.43)$$

7.10 Known Good Convolutional Codes

LO 2

In this section we shall look at some known good convolutional codes. So far only a few constructive classes of convolutional codes have been reported. There exists no class with an algebraic structure comparable to the t -error correcting BCH codes. No constructive methods exist for finding convolutional codes of long constraint length. Most of the codes presented here have been found by computer searches.

Initial work on short convolutional codes with maximal free distance was reported by Odenwalder (1970) and Larsen (1973). A few of the codes are listed in Tables 7.2, 7.3 and 7.4 for code rates 1/2, 1/3 and 1/4 respectively. Table 7.5 lists rate $\frac{1}{2}$ codes used in wireless communication standards. The generator is given as the sequence $1, \gamma_1^{(i)}, \gamma_2^{(i)}, \dots$ where

$$g_i(D) = 1 + \gamma_1^{(i)} D + \gamma_2^{(i)} D^2 + \dots + \gamma_{K-1}^{(i)} D^{K-1} \quad (7.44)$$

For example, the octal notation for the generators of the $R = \frac{1}{2}$, $v = 4$ encoders are 15 and 17 (see Table 7.2). The octal 15 can be deciphered as $15 = 1\text{-}5 = 1\text{-}101$. Therefore,

$$g_1(D) = 1 + (1)D + (0)D^2 + (1)D^3 = 1 + D + D^3$$

Similarly, $17 = 1\text{-}7 = 1\text{-}111$. Therefore,

$$g_2(D) = 1 + (1)D + (1)D^2 + (1)D^3 = 1 + D + D^2 + D^3, \text{ and}$$

$$\mathbf{G}(D) = [1 + D + D^3 \quad 1 + D + D^2 + D^3]$$

Table 7.2 Rate 1/2 codes with maximum free distance.

DID YOU KNOW ?		v	n	Generators (octal)	d_{free}	Heller Bound
Non-Catastrophic	3	6	5	7	5	5
	4	8	15	17	6	6
	5	10	23	35	7	8
	6	12	53	75	8	8
	7	14	133	171	10	10
Catastrophic	5	10	27	35	8	8
	12	24	5237	6731	16	16
	14	28	21645	37133	17	17

Table 7.3 Rate 1/3 codes with maximum free distance

	v	n	Generators (octal)			d_{free}	Heller Bound
Rate 1/3	3	9	5	7	7	8	8
	4	12	13	17	17	10	10
	5	15	25	37	37	12	12
	6	18	47	75	75	13	13
	7	21	133	175	175	15	15

Table 7.4 Rate 1/4 codes with maximum free distance

	v	n	Generators (octal)				d_{free}	Heller Bound
Rate 1/4	3	12	5	7	7	7	10	10
	4	16	13	15	15	17	15	15
	5	20	25	27	33	37	16	16
	6	24	53	67	71	75	18	18
	7	28	135	135	147	163	20	20

INDUSTRIAL RELEVANCE

**Table 7.5** Some rate 1/2 codes used in wireless communication standards

	v	Generators (octal)		d_{free}
GSM	4	31	33	7
802.11a	6	155	117	10
802.11b	6	133	175	9
IS-95	8	657	435	12

Example 7.18 In an Automatic Repeat reQuest (ARQ) scheme, whenever an error is detected in the transmitted message, a retransmission request is sent to the transmitter. ARQ schemes become less efficient (longer delays) as channel condition worsens (low SNR). This behaviour is unacceptable for delay-sensitive applications such as the digital voice communications. One way to address this problem is to use **Hybrid FEC/ARQ** schemes. Convolutional codes are commonly used as the FEC scheme. There are two types of hybrid FEC/ARQ schemes.

In a **type-I hybrid FEC/ARQ** scheme, the message and error detecting parity bits generated by the ARQ system are further encoded with the FEC code. At the receiver, the error correction parity bits are used to correct channel errors.

In a **type-II hybrid FEC/ARQ** scheme, the first transmission of a message is coded with error detection parity bits alone, as in a standard ARQ protocol. If the receiver detects errors in the received block, it saves the erroneous block in a buffer and requests for a retransmission. The retransmitted information is a block of parity bits derived by applying an FEC code to the message. The receiver uses these parity bits to correct the block stored in the receiver buffer.

We will next study an interesting class of codes, called Turbo codes, which lie somewhere between linear block codes and convolutional codes.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 2:

1. Consider the trellis shown in Fig. 7.34. Find the d_{free} .

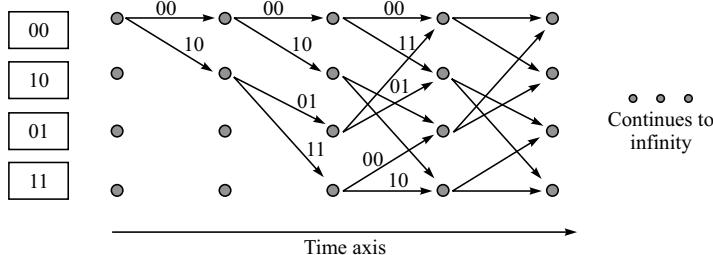


Fig. 7.34

2. Consider a trellis with $T(D) = 2D^6 + 8D^8 + 14D^{10} + \dots$. What is the free distance, d_{free} , for this code? What is the number of paths with weight d_{free} ?

3. Consider two trellises with

$$T_1(D) = 2D^6 + 8D^8 + 14D^{10} + \dots \text{ and}$$

$$T_2(D) = 4D^6 + 8D^8 + 12D^{10} + \dots$$

Which trellis code should I prefer to use and why?

4. Consider the convolutional encoder given in Fig. 7.35.

Let the received word (with errors) be $r = 10\ 11\ 11\ 01\ 10\ 01\ 11\ 10\ \dots$

Find the transmitted sequence.

5. We have seen that cyclic codes can be implemented in hardware using shift registers (Fig. 7.36). Give the trellis representation of the (7, 4) cyclic Hamming code.

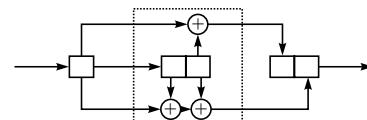


Fig. 7.35

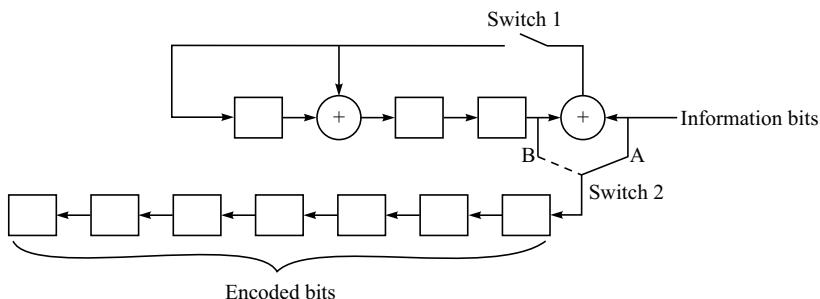


Fig. 7.36

If you have successfully solved the above problems,
you have mastered LO 2. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/613>



7.11 Turbo Codes

LO 3



Underline the concept of Turbo coding and decoding and also interleaver design for Turbo codes.

Turbo Codes were introduced in 1993 at the *International Conference on Communications* (ICC) by Berrou, Glavieux and Thitimajshima in their paper ‘Near Shannon Limit Error Correction Coding and Decoding—Turbo-codes’. In this paper, they quoted a BER performance of 10^{-5} at an E_b/N_0 of 0.7 dB using only a 1/2 rate code, generating tremendous interest in the field. Turbo codes perform well in the low SNR scenario. At high SNRs, some of the traditional codes like the Reed Solomon code can have comparable or better performance than Turbo codes.

Even though Turbo codes are considered as block codes, they do not exactly work like block codes. Turbo codes are actually a quasi-mix between Block and Convolutional codes. They require, like a block code, that the whole block be present before encoding can begin. However, rather than computing parity bits from a system of equations, they use shift registers just like convolutional codes. Turbo encoders typically use at least two convolutional component encoders separated by an **Interleaver**. This is known as **concatenation**. Three different arrangements of turbo codes are Parallel Concatenated Convolutional Codes (PCCC), Serial Concatenated Convolutional Codes (SCCC) and Hybrid Concatenated Convolutional Codes (HCCC). Typically, Turbo codes are arranged like the PCCC. An example of a PCCC Turbo encoder is as follows in Fig. 7.37. As is obvious from Fig. 7.37, the two encoders run in parallel.

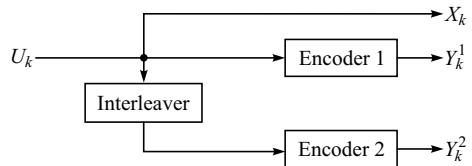


Fig. 7.37 Block diagram of a rate 1/3, PCCC Turbo Encoder.

Definition 7.14 An **Interleaver** π is a permutation $i \rightarrow \pi(i)$ that changes the order of a data sequence of N input symbols d_1, d_2, \dots, d_N .

If the input data sequence is $\mathbf{d} = [d_1, d_2, \dots, d_N]$, then the permuted data sequence is $\mathbf{d}\mathbf{P}$, where \mathbf{P} is an interleaving matrix with a single one in each row and column, all other entries being zero. Every interleaver has a corresponding **Deinterleaver**, π^{-1} , that acts on the interleaved data sequence and restores it to its original order. The deinterleaving matrix is simply the transpose of the interleaving matrix \mathbf{P}^T .

One reason for the better performance of Turbo codes is that they produce high weight codewords. For example, if the input sequence (U_k) is originally low weight, the systematic (X_k) and parity 1 (Y_k^1) outputs may produce a low weight codeword. However, the parity 2 output (Y_k^2) is less likely to be a low weight codeword due to the interleaver in front of it. The interleaver shuffles the input sequence, U_k , in such a way that when introduced to the second encoder, it is more likely to produce a high weight codeword. This is ideal for the code because high weight codewords result in better decoder performance. Intuitively, when one of the encoders produces a ‘weak’ codeword, the other encoder has a low probability of producing another ‘weak’ codeword because of the interleaver. The concatenated version of the two codewords is, therefore, a ‘strong’ codeword. Here, the expression ‘weak’ is used as a measure of the average Hamming distance of a codeword from all other codewords.

Although the encoder determines the *capability* for the error correction, it is the decoder that determines the *actual performance*. The performance, however, depends upon which algorithm is used. Since Turbo decoding is an iterative process, it requires a soft output algorithm like the *maximum a posteriori* algorithm (MAP) or the *Soft Output Viterbi Algorithm* (SOVA) for decoding. Soft output algorithms outperform hard decision algorithms because they have available a better estimate of what the sent data actually was. This

is because soft output yields a gradient of information about the computed information bit rather than just choosing a 1 or 0 like hard output. A typical Turbo decoder is shown in Fig. 7.38.

The MAP algorithm is often used to estimate the most likely information bit to have been transmitted in a coded sequence. The MAP algorithm is favoured because it outperforms other algorithms, such as the SOVA, under low SNR conditions. The major drawback, however, is that it is more complex than most algorithms because of its focus on each individual bit of information. Research in the area (in late 1990s) has resulted in great simplifications of the MAP algorithm.

A Turbo decoder generally uses the MAP algorithm in at least one of its component decoders. The decoding process begins by receiving partial information from the channel (X_k and Y_k^1) and passing it to the first decoder. The rest of the information, parity 2 (Y_k^2), goes to the second decoder and waits for the rest of the information to catch up. While the second decoder is waiting, the first decoder makes an estimate of the transmitted information, interleaves it to match the format of parity 2 and sends it to the second decoder. The second decoder takes information from both the first decoder and the channel and re-estimates the information. This second estimation is looped back to the first encoder where the process starts again. The iterative process of the Turbo decoder is illustrated below in Fig. 7.39.

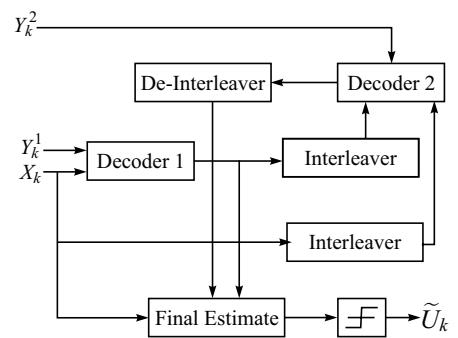


Fig. 7.38 Block diagram of a Turbo decoder.

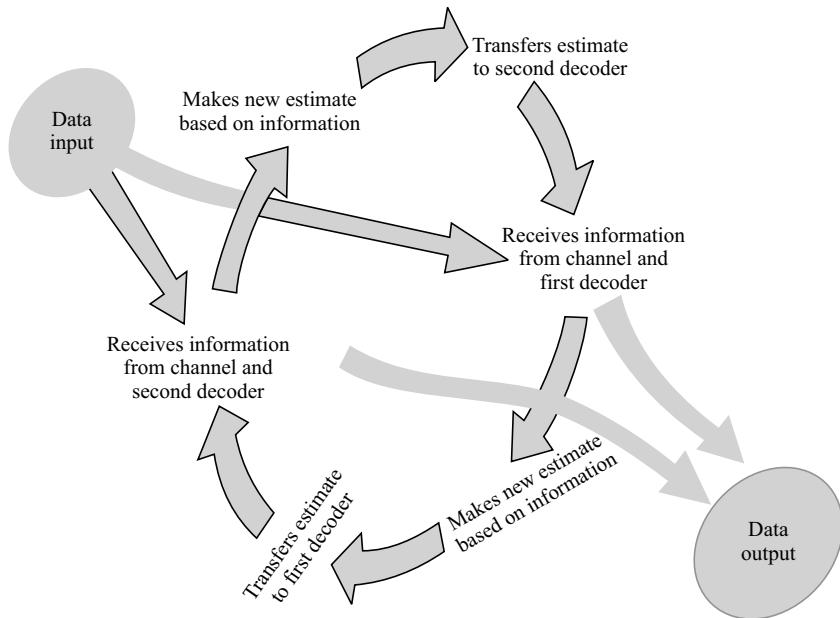


Fig. 7.39 Iterative decoding of Turbo code.

This cycle will continue until certain conditions are met, such as a certain number of iterations are performed. It is from this iterative process that Turbo coding gets its name. The decoder circulates estimates of the sent data like a turbo engine circulates air. When the decoder is ready, the estimated information is

finally kicked out of the cycle and hard decisions are made in the threshold component. The result is the decoded information sequence.

INDUSTRIAL RELEVANCE



Example 7.19 Consider the Turbo encoder shown in Fig. 7.40, which is used in 3GPP-LTE systems. The transfer function of the 8-state constituent code for parallel concatenated convolutional code is

$$G(D) = \begin{bmatrix} 1 & \frac{g_1(D)}{g_0(D)} \end{bmatrix}$$

where, $g_0(D) = 1 + D^2 + D^3$ and $g_1(D) = 1 + D + D^3$.

The initial values of the shift registers of the 8-state constituent encoders are all zeroes when starting to encode the input bits.

The structure of rate 1/3 Turbo encoder is given in Fig. 7.41. Suppose, the input sequence to the Turbo encoder is $X_k = 11001001 \dots$, then the corresponding output for the 1st constituent encoder will be $Z_k = 10000101$. The output Z'_k will depend on the particular interleaver used.

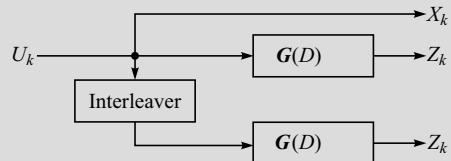


Fig. 7.40 Turbo encoder for 3GPP-LTE.

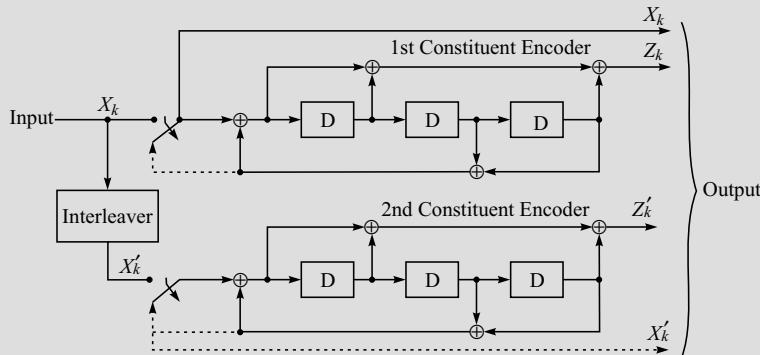


Fig. 7.41 Structure of the Turbo encoder for 3GPP-LTE.

7.12 Turbo Decoding

LO 3

We have seen that the Viterbi algorithm can be used for the decoding of convolutional codes. The Viterbi algorithm performs a systematic elimination of the paths in the trellis. However, such luck does not exist for Turbo decoder. The presence of the interleaver complicates the matter immensely. Before the discovery of Turbo codes, a lot of work was being done in the area of suboptimal decoding strategies for concatenated codes, involving multiple decoders. The symbol-by-symbol *maximum a posteriori* (MAP) algorithm of Bahl, Cocke, Jelinek and Raviv, published in the *IEEE Transactions on Information Theory* in March 1974 also received some attention. It was this algorithm, which was used by Berrou *et al.* in the iterative decoding of their Turbo codes. In this section we shall discuss two methods useful for Turbo decoding:

- (i) The modified Bahl, Cocke, Jelinek and Raviv (BCJR) Algorithm
- (ii) The Iterative MAP decoding.

A. Modified Bahl, Cocke, Jelinek and Raviv (BCJR) Algorithm

The modified BCJR decoding algorithm is a symbol-by-symbol decoder. The decoder decides $u_k = +1$ if

$$P(u_k = +1 | \mathbf{y}) > P(u_k = -1 | \mathbf{y}) \quad (7.45)$$

and it decides $u_k = -1$ otherwise, where $\mathbf{y} = (y_1, y_2, \dots, y_n)$ is the noisy received word.

More succinctly, the decision \hat{u}_k is given by

$$\hat{u}_k = \text{sign}[L(u_k)] \quad (7.46)$$

where $L(u_k)$ is the log *a posteriori* probability (LAPP) ratio defined as

$$L(u_k) = \log \left(\frac{P(u_k = +1 | \mathbf{y})}{P(u_k = -1 | \mathbf{y})} \right) \quad (7.47)$$

Incorporating the code's trellis, this may be written as

$$L(u_k) = \log \left(\frac{\sum_{S^+} p(s_{k-1} = s', s_k = s, \mathbf{y}) / p(\mathbf{y})}{\sum_{S^-} p(s_{k-1} = s', s_k = s, \mathbf{y}) / p(\mathbf{y})} \right) \quad (7.48)$$

where $s_k \in S$ is the state of the encoder at time k , S^+ is the set of ordered pairs (s', s) corresponding to all state transitions $(s_{k-1} = s')$ to $(s_k = s)$ caused by data input $u_k = +1$, and S^- is similarly defined for $u_k = -1$.

Let us define,

$$\gamma_k(s', s) = p(s_k = s, y_k | s_{k-1} = s') \quad (7.49)$$

$$\tilde{\alpha}_k(s) = \frac{\sum_{s'} \alpha_{k-1}(s') \gamma_k(s', s)}{\sum_s \sum_{s'} \alpha_{k-1}(s') \gamma_k(s', s)} \text{ and} \quad (7.50)$$

$$\tilde{\beta}_{k-1}(s') = \frac{\sum_s \tilde{\beta}_k(s) \gamma_k(s', s)}{\sum_s \sum_{s'} \tilde{\alpha}_{k-1}(s') \gamma_k(s', s)} \quad (7.51)$$

with boundary conditions

$$\begin{aligned} \alpha_0(0) &= 1 \text{ and } \alpha_0(s \neq 0) = 0 \\ \beta_N(0) &= 1 \text{ and } \beta_N(s \neq 0) = 0 \end{aligned} \quad (7.52)$$

Then the modified BCJR algorithm gives the LAPP ratio in the following form

$$L(u_k) = \log \left(\frac{\sum_{S^+} \tilde{\alpha}_{k-1}(s') \gamma_k(s', s) \tilde{\beta}_k(s)}{\sum_{S^-} \tilde{\alpha}_{k-1}(s') \gamma_k(s', s) \tilde{\beta}_k(s)} \right) \quad (7.53)$$

B. Iterative MAP decoding

The decoder is shown in Fig. 7.42. D1 and D2 are the two decoders. S is the set of 2^m constituent encoder states. \mathbf{y} is the noisy received word. Using the Baye's rule we can write $L(u_k)$ as

$$L(u_k) = \log \left(\frac{P(\mathbf{y} | u_k = +1)}{P(\mathbf{y} | u_k = -1)} \right) + \log \left(\frac{P(u_k = +1)}{P(u_k = -1)} \right) \quad (7.54)$$

with the second term representing *a priori* information. Since $P(u_k = +1) = P(u_k = -1)$ typically, the *a priori* term is usually zero for conventional decoders. However, for iterative decoders, D1 receives extrinsic or soft information for each u_k from D2 which serves as *a priori* information. Similarly, D2 receives extrinsic information from D1 and the decoding iteration proceeds with the each of the two decoders passing soft information to the other decoder at each half-iteration except for the first. The idea behind extrinsic information is that D2 provides soft information to D1 for each u_k , using only information not available to D1. D1 does likewise for D2.

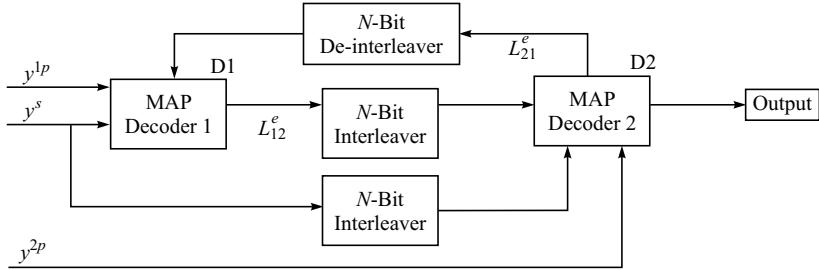


Fig. 7.42 Implementation of the iterative turbo decoder, which uses two MAP decoders operating cooperatively

At any given iteration, D1 computes

$$L_1(u_k) = L_c y_k^s + L_{21}^e(u_k) + L_{12}^e(u_k) \quad (7.55)$$

where, the first term is the channel value, $L_c = 4E_c/N_0$ (E_c = energy per channel bit), $L_{21}^e(u_k)$ is extrinsic information passed from D2 to D1, and $L_{12}^e(u_k)$ is the extrinsic information from D1 to D2.

$L_{12}^e(u_k)$ and $L_{21}^e(u_k)$ are calculated as follows.

$$L_{12}^e(u_k) = \log \left(\frac{\sum_{S^+} \tilde{\alpha}_{k-1}^{(1)}(s') \gamma_k^e(s', s) \tilde{\beta}_k^{(1)}(s)}{\sum_{S^-} \tilde{\alpha}_{k-1}^{(1)}(s') \gamma_k^e(s', s) \tilde{\beta}_k^{(1)}(s)} \right) \quad (7.56)$$

$$L_{21}^e(u_k) = \log \left(\frac{\sum_{S^+} \tilde{\alpha}_{k-1}^{(2)}(s') \gamma_k^e(s', s) \tilde{\beta}_k^{(2)}(s)}{\sum_{S^-} \tilde{\alpha}_{k-1}^{(2)}(s') \gamma_k^e(s', s) \tilde{\beta}_k^{(2)}(s)} \right) \quad (7.57)$$

where

$$\gamma_k^e(s', s) = \exp \left[\frac{1}{2} L_c y_k^p x_k^p \right] \quad (7.58)$$

$$\gamma_k(s', s) = \exp \left[\frac{1}{2} u_k (L^e(u_k) + L_c y_k^s) \right] \cdot \gamma_k^e(s', s) \quad (7.59)$$

$$\tilde{\alpha}_k(s) = \frac{\sum_{s'} \alpha_{k-1}(s') \gamma_k(s', s)}{\sum_s \sum_{s'} \alpha_{k-1}(s') \gamma_k(s', s)}, \text{ and} \quad (7.60)$$

$$\tilde{\beta}_{k-1}(s') = \frac{\sum_{s'} \tilde{\beta}_k(s) \gamma_k(s', s)}{\sum_s \sum_{s'} \tilde{\alpha}_{k-1}(s') \gamma_k(s', s)} \quad (7.61)$$

For the above algorithm, each decoder must have full knowledge of the trellis of the constituent encoders i.e., each decoder must have a table containing the input bits and parity bits for all possible state transitions s' to s . Also, care should be taken that the last m bits of the N -bit information word to be encoded must force encoder1 to the zero state by the N^{th} bit. The performance of iterative decoding improves if the information that is sent to each decoder from the other decoders is less correlated with the input information data sequence.

The complexity of convolutional codes has slowed the development of low-cost TCC decoders. On the other hand, another type of turbo code, known as **Turbo Product Code** (TPC), uses block codes, solving multiple steps simultaneously, thereby achieving high data throughput in the hardware. Here we shall have a brief introduction to product codes. Let us consider two systematic linear block codes C_1 with parameters (n_1, k_1, d_1) and C_2 with parameters (n_2, k_2, d_2) where n_i, k_i and d_i ($i = 1, 2$) stand for codeword length, number of information bits and minimum Hamming distance, respectively. The concatenation of two block codes (or product code) $P = C_1 \times C_2$ is obtained (see Fig. 7.43) by the following steps.

1. placing $(k_1 \times k_2)$ information bits in an array of k_1 rows and k_2 columns,
2. coding the k_1 rows using code C_2 ,
3. coding the n_2 columns using code C_1 .

The parameters of the product code P are : $n = n_1 \times n_2, k = k_1 \times k_2, d = d_1 \times d_2$ and the code rate R is given by $R_1 \times R_2$ where R_i is the code rate of code C_i . Thus, we can build very long block codes with large minimum Hamming distance by combining short codes with small minimum Hamming distance. Given the procedure used to construct the product code, it is clear that the $(n_2 - k_2)$ last columns of the matrix are codewords of C_1 . By using the matrix generator, one can show that the last rows of matrix P are codewords of C_2 .

Hence all the rows of matrix P are codewords of C_1 and all the columns of matrix P are codewords of C_2 .

Let us now consider the decoding of the rows and columns of a product code P transmitted on a Gaussian channel using QPSK signalling. On receiving matrix \mathbf{R} corresponding to a transmitted codeword \mathbf{E} , the first decoder performs the soft decoding of the rows (or columns) of P using as input matrix \mathbf{R} . Soft Input/Soft Output decoding is performed using the new algorithm proposed by R. Pyndiah. By subtracting the soft input from the soft output we obtain the extrinsic information $\mathbf{W}(2)$ where index 2 indicates that we are considering the extrinsic information for the second decoding of P which was computed during the first decoding of P . The soft input for the decoding of the columns (or rows) at the second decoding of P is given by

$$\mathbf{R}(2) = \mathbf{R} + a(2)\mathbf{W}(2) \quad (7.62)$$

where $a(2)$ is a scaling factor which takes into account the fact that the standard deviation of samples in matrix \mathbf{R} and in matrix \mathbf{W} are different. The standard deviation of the extrinsic information is very high in the first decoding steps and decreases as we iterate the decoding. This scaling factor, a , is also used to reduce the effect of the extrinsic information in the soft decoder in the first decoding steps when the BER is relatively high. It takes a small value in the first decoding steps and increases as the BER tends to 0. The decoding procedure described above is then generalised by cascading elementary decoders illustrated in Fig. 7.44.

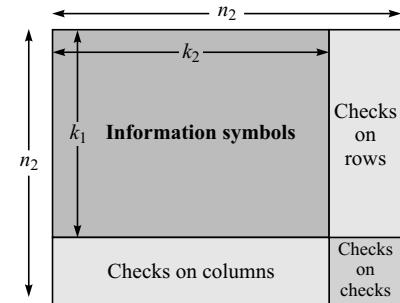


Fig. 7.43 Example of a product code $P = C_1 \times C_2$.

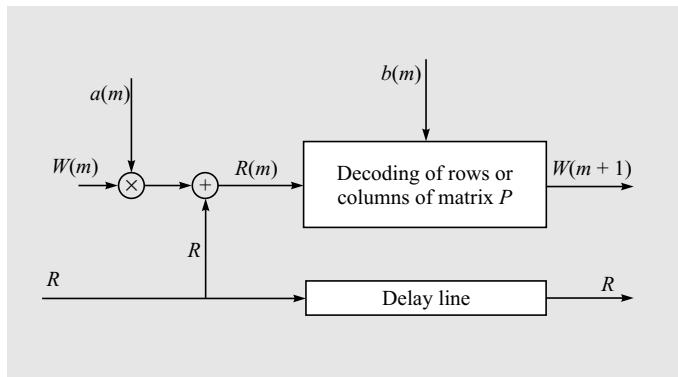


Fig. 7.44 Block diagram of elementary block turbo decoder.

Let us now briefly look at the performance of Turbo codes and compare it to other existing schemes. As shown in Fig. 7.45, Turbo codes are the best practical codes due to their performance at *low* SNR (at high SNRs, the Reed Solomon codes outperform Turbo codes!). It is obvious from the graph that the Recursive Systematic Convolutional (RSC) Turbo code is the best practical code known so far because it can achieve low BER at low SNR and is the closest to the theoretical maximum of channel performance, the Shannon limit. The magnitude of how well it performs is determined by the coding gain. It can be recalled that the coding gain is the difference in SNR between a coded channel and an uncoded channel for the same performance (BER). Coding gain can be determined by measuring the distance between the SNR values of any of the coded channels and the uncoded channel at a given BER. For example, the coding gain for the RSC Turbo code, with rate 1/2 at a BER of 10^{-5} , is about 8.5 dB. The physical consequence can be visualised as follows. Consider space communications where the received power follows the inverse square law ($P_R \propto 1/d^2$). This means that the Turbo coded signal can either be received 2.65 ($= \sqrt{7}$) times farther away than the uncoded signal (at the same transmitting power), or it only requires 1/7 the transmitting power (for the same transmitting distance). Another way of looking at it is to turn it around and talk about portable device battery lifetimes. For instance, since the RSC Turbo coded channel requires only 1/7 the power of the uncoded channel, we can say that a device using a Turbo codec, such as a cell phone, has a battery life 7 times longer than a device without any channel coding.

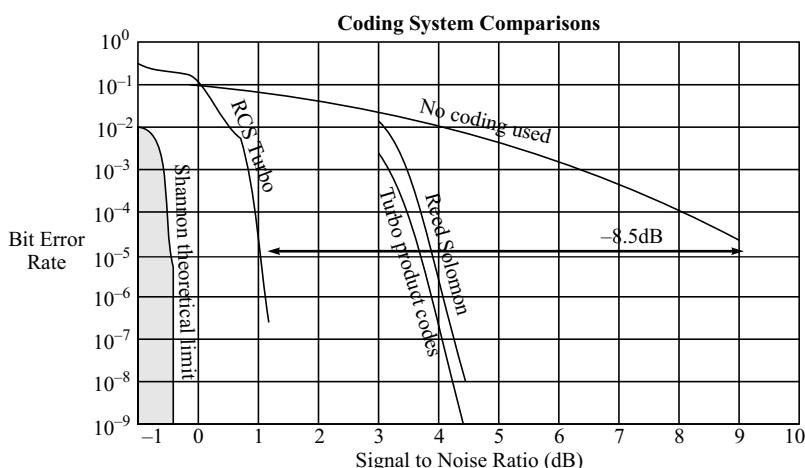


Fig. 7.45 Comparison of different coding systems.

7.13 Interleaver Design for Turbo Codes

LO 3

The superior performance of Turbo codes over convolutional codes is achieved only when the length of the interleaver is very large, to the order of several thousand bits. For large blocklength interleavers, most random interleavers perform well. On the other hand, for some applications, it is preferable to have a deterministic interleaver, to reduce the hardware requirements for interleaving and deinterleaving operations. For short blocklength interleavers, the performance of the Turbo code with a random interleaver degrades substantially up to a point where its bit error rate (BER) performance is worse than the BER performance of convolutional codes with similar computational complexity. For short blocklength interleavers, selection of the type of interleaver has a significant effect on the performance of the Turbo code. In many applications, such as voice, delay is an important issue in choosing the block size. For these applications, there is a need to design short blocklength interleavers that demonstrate acceptable BER performance.

There are two major criteria for the interleaver design.

- (i) The distance spectrum properties (weight distribution) of the code, and
- (ii) The correlation between the soft output of each decoder corresponding to its parity bits and the information input data sequence.

Criterion (ii) is sometimes referred to as the **Iterative Decoding Suitability (IDS)** criterion.

Definition 7.15 A **Random Interleaver** is simply a random permutation, π .

Definition 7.16 An **S-Random Interleaver** (where $S = 1, 2, 3, \dots$) is a ‘semi-random’ interleaver constructed as follows. Each randomly selected integer is compared with S previously selected random integers. If the difference between the current selection and S previous selections is smaller than S , the random integer is rejected. This process is repeated until N distinct integers have been selected.

INDUSTRIAL RELEVANCE



Example 7.20 The Third Generation Partnership Project (3GPP) proposes turbo code interleaver algorithm comprising of a series of mathematically complex processes that map an input sequence of length K ($40 \leq K \leq 5114$) to an interleaved sequence. The algorithm can be summarised as follows.

Step 1: Row-wise data input to an $R \times C$ rectangular matrix, with zero padding if $K < R \times C$.

Step 2: Intra-row permutation of the rectangular matrix, based on a recursively constructed base sequence s .

Step 3: Inter-row permutation of the rectangular matrix, based on a well-defined inter-row permutation pattern T and a permuted least primes sequence r (obtained from T and an ordered least primes sequence q).

Step 4: Column-wise data output from the rectangular matrix, with pruning.

Example 7.21 NASA's Mars Reconnaissance Orbiter (MRO) telecommunications subsystem uses Turbo codes for bit rates above 32 kbps. Turbo codes with rate 1/2, 1/3 and 1/6 are used depending on the channel conditions. This capability, implemented in hardware, provides more link margin as compared to convolutional codes of the same code rate.

Digital Video Broadcasting-Return Channel via Satellite (DVB-RCS) also uses Turbo codes.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 3:

1. Consider the Turbo encoder for in 3GPP-LTE given in Fig. 7.40. Let the interleaver be 'flip all bits'. Encode the bit stream: 0 0 1 1 0 1 1 0 ... using the Turbo encoder. M
2. Again, consider the Turbo encoder for in 3GPP-LTE given in Fig. 7.40. Let the interleaver be 'first in last out' of size 8 bits. Encode the bit stream: 1 0 0 0 1 0 0 0 ... using the Turbo encoder. M

If you have successfully solved the above problems,
you have mastered LO 3. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/614>



7.14 Concluding Remarks

The notion of convolutional codes was first proposed by Elias (1954) and later developed by Wozencraft (1957) and Ash (1963). A class of multiple error correcting convolutional code was suggested by Massey (1963). The study of the algebraic structure of convolutional codes was carried out by Massey (1968) and Forney (1970).

Viterbi decoding was developed by Andrew J. Viterbi, a founder of Qualcomm Corporation. His seminal paper on the technique titled 'Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm,' was published in *IEEE Transactions on Information Theory*, Volume IT-13, pages 260-269, in April, 1967. In 1968, Heller showed that the Viterbi algorithm is practical if the constraint length is not too large.

Turbo codes represent the next leap forward in error correction. Turbo codes were introduced in 1993 at the *International Conference on Communications* (ICC) by Berrou, Glavieux and Thitimajshima in their paper 'Near Shannon Limit Error Correction Coding and Decoding - Turbo-codes'. These codes get their names from the fact that the decoded data are recycled through the decoder several times. The inventors probably found this reminiscent of the way a turbocharger operates. Turbo codes have been shown to perform within 1 dB of the Shannon limit at a BER of 10^{-7} . They break a complex decoding problem down into

simple steps, where each step is repeated until a solution is reached. The term ‘turbo code’ is often used to refer to turbo convolutional codes (TCCs) — one form of turbo codes. The symbol-by-symbol maximum *a posteriori* (MAP) algorithm of Bahl, Cocke, Jelinek and Raviv, published in 1974 (nineteen years before the introduction of Turbo codes!), was used by Berrou *et al.* for the iterative decoding of their Turbo codes. The complexity of convolutional codes has slowed the development of low-cost TCC decoders. On the other hand, another type of turbo code, known as turbo product code (TPC), uses block codes, solving multiple steps simultaneously, thereby achieving high data throughput in hardware.

LEARNING OUTCOMES

- An important sub-class of tree codes is called convolutional codes. Convolutional codes make decisions based on past information, i.e., memory is required. A (n_0, k_0) tree code that is linear, time-invariant and has a finite word length $k = (m + 1)k_0$ is called an (n, k) convolutional code.
- For convolutional codes, much smaller blocks of uncoded data of length k_0 are used. These are called information frames. These information frames are encoded into codeword frames of length n_0 . The rate of this tree code is defined as $R = \frac{k_0}{n_0}$.
- The constraint length of a shift register encoder is defined as the number of symbols it can store in its memory.
- For convolutional codes, the generator polynomial matrix of size $k_0 \times n_0$ for a convolutional code is given by $\mathbf{G}(D) = [g_{ij}(D)]$, where, $g_{ij}(D)$ are the generator polynomials of the code. $g_{ij}(D)$ are obtained simply by tracing the path from input i to output j .
- The word length of a convolutional code is given by $k = k_0 \max_{i,j} [\deg g_{ij}(D) + 1]$, the blocklength is given by $n = n_0 \max_{i,j} [\deg g_{ij}(D) + 1]$ and the constraint length is given by $v = \sum_{i=1}^{k_0} \max_j [\deg g_{ij}(D)]$.
- The encoding operation can simply be described as vector matrix product, $\mathbf{C}(D) = \mathbf{I}(D)\mathbf{G}(D)$, or equivalently, $c_j(D) = \sum_{l=1}^{k_0} i_l(D)g_{l,j}(D)$.
- A parity check matrix $\mathbf{H}(D)$ is an $(n_0 - k_0)$ by n_0 matrix of polynomials that satisfies $\mathbf{G}(D)\mathbf{H}(D)^T = \mathbf{0}$, and the syndrome polynomial vector which is a $(n_0 - k_0)$ -component row vector is given by $\mathbf{s}(D) = \mathbf{v}(D)\mathbf{H}(D)^T$.
- A systematic encoder for a convolutional code has the generator polynomial matrix of the form $\mathbf{G}(D) = [\mathbf{I} | \mathbf{P}(D)]$, where \mathbf{I} is a k_0 by k_0 identity matrix and $\mathbf{P}(D)$ is a k_0 by $(n_0 - k_0)$ matrix of polynomials. The parity check polynomial matrix for a systematic convolutional encoder is $\mathbf{H}(D) = [-\mathbf{P}(D)^T | \mathbf{I}]$.
- Two generator polynomial matrices (transfer functions), $\mathbf{G}_1(D)$ and $\mathbf{G}_2(D)$, are said to be equivalent if they generate the same convolutional code. For two equivalent convolutional encoders, $\mathbf{G}_2(D) = \mathbf{A}(D)\mathbf{G}_1(D)$ for an invertible matrix, $\mathbf{A}(D)$.
- A convolutional code whose generator polynomials $g_1(D), g_2(D), \dots, g_{n_0}(D)$ satisfy $\text{GCD}[g_1(D), g_2(D), \dots, g_{n_0}(D)] = x^a$, for some a is called a non-catastrophic convolutional code. Otherwise it is called a catastrophic convolutional code.

- The l^{th} minimum distance d_l^* of a convolutional code is equal to the smallest Hamming distance between any two initial codeword segments l frame long that are not identical in the initial frame. If $l = m + 1$, then this $(m + 1)^{\text{th}}$ minimum distance is called the minimum distance of the code and is denoted by d^* , where m is the number of information frames that can be stored in the memory of the encoder. In literature, the minimum distance is also denoted by d_{\min} .
- If the l^{th} minimum distance of a convolutional code is d_l^* , the code can correct t errors occurring in the first l frames provided, $d_l^* \geq 2t + 1$. The free distance of a convolutional code is given by $d_{\text{free}} = \max_l [d_l]$.
- The free length n_{free} of a convolutional code is the length of the non-zero segment of a smallest weight convolutional codeword of non-zero weight. Thus, $d_l = d_{\text{free}}$ if $l = n_{\text{free}}$, and $d_l < d_{\text{free}}$ if $l < n_{\text{free}}$. In literature, n_{free} is also denoted by n_{∞} .
- Another way to find out the d_{free} of a convolutional code is to use the concept of a generating function, whose expansion provides all the distance information directly.
- The generator matrix for the convolutional code is given by

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_2 & \dots & \mathbf{G}_m & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{G}_0 & \mathbf{G}_1 & \dots & \mathbf{G}_{m-1} & \mathbf{G}_m & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{0} & \mathbf{G}_0 & \dots & \mathbf{G}_{m-2} & \mathbf{G}_{m-1} & \mathbf{G}_m & \mathbf{0} & \mathbf{0} & \dots \\ \vdots & \vdots & & & & & & & & \vdots \end{bmatrix}.$$

- The Viterbi decoding technique is an efficient decoding method for convolutional codes. Its computational requirements grow exponentially as a function of the constraint length.
- For rate R and constraint length, let d be the largest integer that satisfies $H\left(\frac{d}{n_0 v}\right) \leq 1 - R$. Then at least one binary convolutional code exists with minimum distance d for which the above inequality holds. Here $H(x)$ is the familiar entropy function for a binary alphabet.
- For a binary code with $R = 1/n_0$ the minimum distance d_{\min} satisfies $d_{\min} \leq \lfloor (n_0 v + n_0)/2 \rfloor$, where $\lfloor I \rfloor$ denotes the largest integer less than or equal to I .
- An upper bound on d_{free} is given by Heller is $d_{\text{free}} = \min_{j \geq 1} \left| \frac{n_0}{2} \frac{2^j}{2^j - 1} (v + j - 1) \right|$. To calculate the upper bound, the right hand side should be plotted for different integer values of j . The upper bound is the minimum of this plot.
- For convolutional codes, the upper bound on the first error probability can be obtained by $P_e \leq T(D)|_{D=2\sqrt{p(1-p)}}$ and the bit error probability $P_b \leq \frac{1}{k} \frac{\partial T(D, I)}{\partial I}|_{I=1, D=2\sqrt{p(1-p)}}$.
- Turbo codes are actually a quasi-mix between Block and Convolutional codes. Turbo codes typically use at least two convolutional component encoders and two maximum a posteriori (MAP) algorithm component decoders in the Turbo codec. Although the encoder determines the capability for the error correction, it is the decoder that determines the actual performance.

- For short blocklength interleavers, the selection of the interleaver has a significant effect on the performance of the Turbo code.

It's kind of fun to do the impossible.

Walt Disney (1901-1966)



MULTIPLE CHOICE QUESTIONS

- 7.1 Convolutional codes are a sub-class of

 - (a) Branch codes
 - (b) Nonlinear Trellis codes
 - (c) Tree codes
 - (d) None of the above

7.2 The constraint length of a shift register encoder is defined as

 - (a) The number of input symbols in one information frame
 - (b) The number of symbols it can store in its memory
 - (c) The number of symbols in one codeword frame
 - (d) None of the above

7.3 For the generator polynomial matrix $\mathbf{G}(D) = [g_{ij}(D)]$, where, $g_{ij}(D)$ are the generator polynomials of the code, $g_{ij}(D)$ are obtained by

 - (a) Tracing the path from input i to output j
 - (b) Tracing the path from input j to output i
 - (c) Tracing the path from input i to memory element j
 - (d) None of the above

7.4 The encoding operation for convolutional codes can be described as

 - (a) $\mathbf{C}(D) = \mathbf{I}(D)\mathbf{G}^T(D)$
 - (b) $\mathbf{C}(D) = \frac{\mathbf{I}(D)}{\mathbf{G}(D)}$
 - (c) $\mathbf{C}(D) = \frac{\mathbf{G}(D)}{\mathbf{I}(D)}$
 - (d) $\mathbf{C}(D) = \mathbf{I}(D)\mathbf{G}(D)$

7.5 For a systematic convolutional encoder $\mathbf{G}(D) = [\mathbf{I} | \mathbf{P}(D)]$, the parity check polynomial matrix is given by

 - (a) $\mathbf{H}(D) = [\mathbf{P}(D)^T | \mathbf{I}]$
 - (b) $\mathbf{H}(D) = [-\mathbf{P}(D)^T | \mathbf{I}]$
 - (c) $\mathbf{H}(D) = [-\mathbf{P}(D) | \mathbf{I}]$
 - (d) None of the above

7.6 For two equivalent convolutional encoders, $\mathbf{G}_2(D) = A(D)\mathbf{G}_1(D)$, $A(D)$ should be

 - (a) An invertible matrix
 - (b) An orthogonal matrix
 - (c) A Vandermonde matrix
 - (d) None of the above

7.7 For a non-catastrophic convolutional code with generator polynomials $g_1(D), g_2(D), \dots, g_{n_0}(D)$ we have

 - (a) $\text{GCD}[g_1(D), g_2(D), \dots, g_{n_0}(D)] = x^\alpha$ for some α
 - (b) $\text{GCD}[g_1(D), g_2(D), \dots, g_{n_0}(D)] = 1$

For interactive quiz with answers, scan the QR code given here

or

Visit <http://qrcode.flipick.com/index.php/572>



SHORT-ANSWER TYPE QUESTIONS



- 7.1 Convolutional codes are linear. True or false?

7.2 For the same input bit stream, a convolutional encoder gives the same encoded bit stream as the output. True or false?

7.3 The XOR gates used for circuit implementation of convolutional codes perform addition over $GF(2)$. True or false?

7.4 Find the rate of the encoder with $\mathbf{G}(D) = [D^5 + D^3 + 1 \quad D^4 + 1 \quad D^3 + 1]$.

7.5 Find the rate of the encoder with $\mathbf{G}(D) = \begin{bmatrix} 1 & D^2 & D \\ D & 1 & 0 \end{bmatrix}$.

7.6 Give an example of $R = 1/3$ systematic convolutional encoder with 2 memory units.

7.7 The encoder given by $\mathbf{G}(D) = [1 \quad D^2 + 1 \quad D + 1]$ is catastrophic. True or false?

7.8 For what input stream would the encoder given Example 7.3 generate the output stream 11 11 11 11 ...?

7.9 Any t error correcting convolutional code will also correct a burst error of length t . True or false?

7.10 If $\mathbf{G}(D) = [1 \quad D^6 + D^5 + D^2 + 1]$ can correct 2 errors, how many errors can $\mathbf{G}'(D) = [1 \quad D^{24} + D^{20} + D^8 + 1]$ correct?

7.11 Consider the trellis diagram in Fig. 7.24. Guess the transmitted sequence if the received sequence is $r = 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \dots$

7.12 The constituent encoders for the turbo codes must be identical. True or false?

7.13 Turbo codes get their name because of their similarity of the feedback loop (used for decoding) and the exhaust feedback used for engine turbo charging. True or false?

7.14 Turbo codes are actually a quasi-mix between block and convolutional codes. True or false?

- 7.15 An interleaver is a permutation that changes the order of a data sequence of the input symbols. True or false?

For answers, scan the QR code given here

Or

Visit <http://qrcode.flipick.com/index.php/615>



PROBLEMS



- M** 7.1 Consider the convolutional encoder shown in Fig. 7.46.

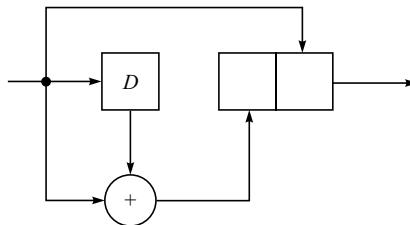


Fig. 7.46

- (i) Draw the state diagram for this encoder.
- (ii) Draw the trellis diagram for this encoder.
- (iii) Use the Viterbi algorithm to decode the sequence $r = 11\ 01\ 10\ 10\ 01\ \dots$

- S** 7.2 Draw the convolutional encoder given by $\mathbf{G}(D) = [1 + D + D^2 \quad 1 + D^2]$.
- S** 7.3 Draw the convolutional encoder given by $\mathbf{G}(D) = \begin{bmatrix} 1 & D & 0 \\ D^2 & 1 & D \end{bmatrix}$.
- S** 7.4 Consider the convolutional encoder given by $\mathbf{G}(D) = [1 + D^2 \quad 1 + D + D^2]$. Is this catastrophic? Find $\mathbf{G}^{-1}(D)$.
- M** 7.5 Consider the convolutional encoder given by $\mathbf{G}(D) = [1 + D^2 \quad 1 + D + D^2 + D^3]$. How many states does this encoder have? Is this catastrophic?

- S** 7.6 Given $\mathbf{G}(D) = \begin{bmatrix} 1 & 0 & \frac{D^2}{1+D^3} \\ 0 & 1 & \frac{D}{1+D^3} \end{bmatrix}$ find $\mathbf{H}(D)$.

- M** 7.7 Draw the trellis diagram for the convolutional code given by $\mathbf{G}(D) = \begin{bmatrix} 1 & D & 0 \\ D^2 & 1 & D \end{bmatrix}$.
- M** 7.8 Consider a rate 1/3 convolutional code with the generator polynomial matrix in octal form given by [6 5 7].
- (i) Draw the circuit of the encoder.

- (ii) Give the generator matrix, \mathbf{G} .
 (iii) Encode the input sequence 1 1 1 0 1 ...
- D** 7.9 Again, consider the (3, 1) convolutional code with the generator polynomial matrix in octal form given by [6 5 7].
 (i) Draw the modified state diagram and determine d_{free} .
 (ii) Use the augmented generating function to determine the length of the path corresponding to d_{free} .
- M** 7.10 Consider a rate 2/3 convolutional code with the parity check polynomial matrix in octal form given by [05 23 27]. Draw the circuit of the encoder.
- D** 7.11 Design a rate $\frac{1}{2}$ convolutional encoder with a constraint length $v = 4$ and $d^* = 6$.
 (i) Construct the state diagram for this encoder.
 (ii) Construct the trellis diagram for this encoder.
 (iii) What is the d_{free} for this code?
 (iv) Give the generator matrix, \mathbf{G} .
 (v) Is this code non-catastrophic? Why?
- D** 7.12 Design a (12, 3) systematic convolutional encoder with a constraint length $v = 3$ and $d^* \geq 8$.
 (i) Construct the trellis diagram for this encoder.
 (ii) What is the d_{free} for this code?
- D** 7.13 Consider the binary encoder shown in Fig. 7.47.
 (i) Construct the trellis diagram for this encoder.
 (ii) Write down the values of k_0, n_0, v, m and R for this encoder.
 (iii) What are the values of d^* and d_{free} for this code?
 (iv) Give the generator polynomial matrix for this encoder.

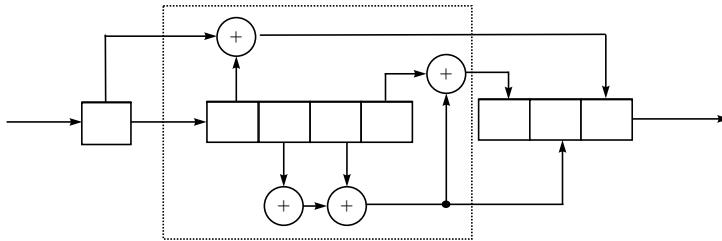


Fig. 7.47

- M** 7.14 Consider the binary encoder shown in Fig. 7.48.
 (i) Write down the values of k, n, v, m and R for this encoder.
 (ii) Give the generator polynomial matrix $\mathbf{G}(D)$ for this encoder.
 (iii) Give the generator matrix \mathbf{G} for this encoder.
 (iv) Give the parity check matrix \mathbf{H} for this encoder.
 (v) What are the values of d^*, d_{free} and n_{free} for this code?
 (vi) Is this encoder optimal in the sense of the Heller bound on d_{free} ?
 (vii) Encode the following sequence of bits using this encoder: 101 001 001 010 000.

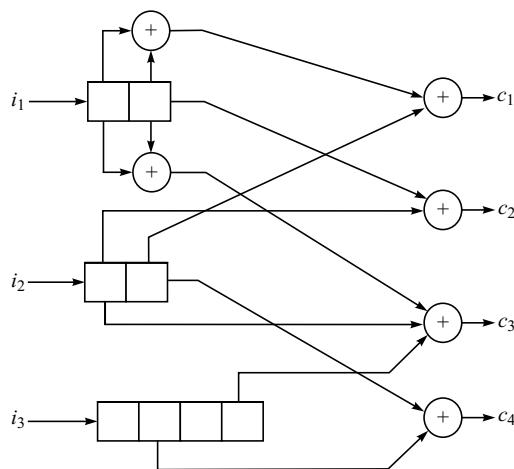


Fig. 7.48

- M 7.15 Consider a convolutional encoder described by its generator polynomial matrix, defined over $GF(2)$:

$$\boldsymbol{G}(D) = \begin{bmatrix} D & 0 & 1 & D^2 & D+D^2 \\ D^2 & 0 & 0 & 1+D & 0 \\ 1 & 0 & D^2 & 0 & D^2 \end{bmatrix}$$

- (i) Draw the circuit realization of this encoder using shift registers. What is the value of v ?
 - (ii) Is this a catastrophic code? Why?
 - (iii) Is this code optimal in the sense of the Heller bound on d_{free} ?

- M** 7.16 The parity check matrix of the $(12, 9)$ Wyner-Ash code for $m = 2$ is given below.

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

- (i) Determine the generator matrix, \mathbf{G} .
 - (ii) Determine the generator polynomial matrix, $\mathbf{G}(D)$.
 - (iii) Give the circuit realization of the $(12, 9)$ Wyner-Ash convolutional code.
 - (iv) What are the values of d^* and d_{free} for this code?

- M** 7.17 Consider a convolutional encoder defined over $GF(4)$ with the generator polynomials

$$g_1(D) = 2D^3 + 3D^2 + 1 \text{ and} \\ g_2(D) = D^3 + D + 1$$

- (i) What is the minimum distance of this code?
 - (ii) Is this code non-catastrophic? Why?

- M** 7.18 Let the generator polynomials of a 1/3 binary convolutional encoder be given by

$$\begin{aligned}g_1(D) &= D^3 + D^2 + 1, \\g_2(D) &= D^3 + D \text{ and} \\g_3(D) &= D^3 + 1.\end{aligned}$$

- (i) Encode the bit stream: 01100011110101.
- (ii) Encode the bit stream: 1010101010 ...
- (iii) Decode the received bit stream: 001001101111000110011.

- M** 7.19 Consider a rate 1/2 convolutional encoder defined over $GF(3)$ with the generator polynomials

$$\begin{aligned}g_1(D) &= 2D^3 + 2D^2 + 1 \text{ and} \\g_2(D) &= D^3 + D + 2.\end{aligned}$$

- (i) Show the circuit realization of this encoder.
- (ii) What is the minimum distance of this code?
- (iii) Encode the following string of symbols using this encoder: 2012111002102.
- (iv) Suppose the error vector is given by 0010102000201. Construct the received vector and then decode this received vector using the Viterbi algorithm.

- D** 7.20 Consider the convolutional encoder over $GF(4)$ shown in Fig. 7.49. The elements of the field are $\{0, 1, \alpha, \alpha^2\}$.

- (i) Show that the code is linear over $GF(4)$.
- (ii) Draw the trellis diagram for the encoder.
- (iii) Is this encoder catastrophic?
- (iv) Find d_{free} for this code.

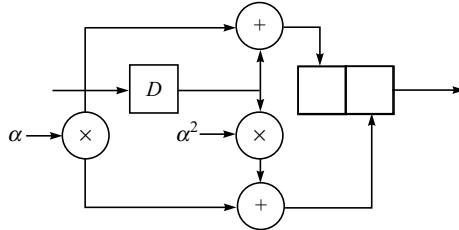


Fig. 7.49

COMPUTER PROBLEMS



- 7.1 Write a computer program that determines the Heller bound on d_{free} , given the values for n_0 and v .
- 7.2 Write a computer program to exhaustively search for good systematic convolutional codes. The program should loop over the parameters k_0 , n_0 , v , m etc. and determine the generator polynomial matrix (in octal format) for the best convolutional code in its category.
- 7.3 Write a program that calculates the d^* and d_{free} given the generator polynomial matrix of any convolutional encoder.

7.4 Write a computer program that constructs all possible rate $\frac{1}{2}$ convolutional encoder for a given constraint length, v and chooses the best code for a given value of v . Using the program, obtain the following plots.

- (i) the minimum distance, d^* versus v , and
- (ii) the free distance, d_{free} versus v .

Comment on the error correcting capability of convolutional codes in terms of the memory requirement.

7.5 Write a Viterbi decoder in software that takes in the following.

- (i) code parameters in the Octal format, and
- (ii) the received bit stream.

The decoder then produces the survivors and the decoded bit stream.

7.6 Verify the Heller bound on the entries in Table 7.4 for $v = 3, 4, \dots, 7$.

7.7 Write a generalised computer program for a Turbo Encoder. The program should take in the parameters for the two encoders and the type of interleaver. It should then generate the encoded bit-stream when an input (uncoded) bit-stream is fed into the program.

7.8 Modify the Turbo Encoder program developed in the previous question to determine the d_{free} of the Turbo encoder.

7.9 Consider a rate 1/3 Turbo Encoder shown in Fig. 7.50. Let the random interleaver size be 256 bits.

- (i) Find the d_{free} of this Turbo encoder.
- (ii) If the input bit rate is 28.8 kb/s, what is the time delay caused by the encoder.

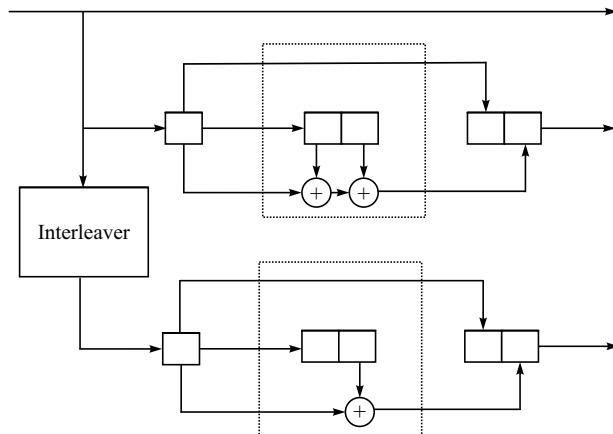


Fig. 7.50

7.10 Write a generalised computer program that performs Turbo decoding using the iterative MAP decoding algorithm. The program should take in the parameters for the two encoders, the type of interleaver used for encoding and the SNR. It should produce a sequence of decoded bits when fed with a noisy, encoded bit-stream.

7.11 We wish to find the performance of the Turbo encoder shown in Fig. 7.50 over AWGN channels. Generate the BER versus SNR curves for an S -random interleaver. Run your simulations for interleaver sizes of 64, 256 and 1024. Plot the family of BER versus SNR curves for $S = 5, 10$ and 15. Use the iterative MAP decoding algorithm for this problem.

PROJECT IDEAS

7.1 Iwadare Code: For positive integers λ and n_0 , an Iwadare code is defined as

$$\mathbf{G}(D) = \begin{bmatrix} 1 & & g_1(D) \\ & 1 & g_2(D) \\ & \ddots & \vdots \\ & & 1 & g_{n_0-1}(D) \end{bmatrix}$$

where $g_i(D) = D^{(\lambda+1)(2n_0-i)+i-3} + D^{(\lambda+1)(n_0-i)-1}$ for $i = 1, 2, \dots, n_0-1$. It is a $((m+1)n_0, (m+1)(n_0-1))$ code. Implement the Iwadare encoder in software and also give a hardware realisation for the $(72, 48)$ Iwadare code. What is the burst error capability of the Iwadare code?

7.2 Turbo Encoder and Decoder: Consider the rate $1/3$ Turbo encoder comprising the following constituent encoders.

$$G_1(D) = G_2(D) = \left[1 \quad \frac{1+D^2+D^3+D^4}{1+D+D^4} \right]$$

The encoded output consists of the information bit, followed by the two *parity* bits from the two encoders. Thus the rate of the encoder is $1/3$. Use a random interleaver of size 257.

- (i) For this Turbo encoder, generate a plot for the bit error rate (BER) versus the signal to noise ratio (SNR). Vary the SNR from -2 dB through 10 dB.
- (ii) Repeat the above for an interleaver of size 1024. Comment on your results.

REFERENCES FOR FURTHER READING

1. Roth, R.M., *Introduction to Coding Theory*, Cambridge University Press, 2006.
2. Lin, S. and Costello, D.J., *Error Control Coding: Fundamentals and Applications* (2nd edition), Prentice-Hall, 2004.
3. Johannesson, R. and Zigangirov, K.S., *Fundamentals of Convolutional Coding* (2nd ed.), Wiley-IEEE Press, 2015.
4. Blahut, R.E., *Algebraic Codes for Data Transmission*, Cambridge University Press, 2002.

Trellis Coded Modulation

A mathematical theory is not to be considered complete until you have made it so clear that you can explain it to the first man whom you meet on the street.

David Hilbert (1862-1943)

Learning objectives

After studying this chapter the students will be able to:

- LO 1** Explain the concept of Trellis Coded Modulation (TCM) and carry out encoding using a TCM scheme.
- LO 2** Know how to perform TCM decoding and design TCM schemes for AWGN and Fading channels.
- LO 3** Underline the concept of Space Time Trellis Codes (STTC) and learn the STTC design criteria for fading channels.

8.1 Introduction to TCM

In the previous chapters we have studied a number of error control coding techniques. In all these techniques, extra bits are added to the information bits in a *known* manner. However, the improvement in the bit error rate is obtained at the expense of bandwidth, because of these extra bits. This bandwidth expansion is equal to the reciprocal of the code rate. For example, a RS (255, 223) code has a code rate, $R = 223/255 = 0.8745$ and $1/R = 1.1435$. Hence, to send 100 information bits, we have to transmit 14.35 extra bits (overhead). This translates to a bandwidth expansion of 14.35%. Even for this efficient RS (255, 223) code the excess bandwidth requirement is not small. In power limited channels (like deep space communications) one may trade the bandwidth expansion for a desired performance. However, for bandwidth limited channels (like the telephone channel), this may not be an easy option. In such channels a bandwidth efficient signalling scheme such as Pulse Amplitude Modulation (PAM), Quadrature Amplitude Modulation (QAM) or Multi Phase Shift Keying (MPSK) is usually employed to support high bandwidth

...
This chapter comes with a video overview by the author. Scan here to know more or
[Visit `http://qrcode.flipick.com/index.php/621`](http://qrcode.flipick.com/index.php/621)



efficiency (in bits/s/Hz). In general, either the bandwidth (data rate) or the power must be sacrificed in order to improve the performance (error rate). Is it possible to achieve an improvement without sacrificing either the data rate or the bandwidth without using additional power? In this chapter we will study another coding technique, called the **Coded Modulation Technique**, which can achieve a better performance *without* the bandwidth expansion or using extra power. At the end of the chapter, we will also briefly study **Space Time Trellis Codes** in which the branches of the trellis are labelled by the signals transmitted by the multiple antennas.

In this chapter

We will start with the concept of Trellis Coded Modulation (TCM). We will then learn how to encode an input sequence using a TCM scheme. We will define the asymptotic coding gain. We will then introduce the basic idea behind set partitioning followed by Ungerboek's design criteria, in LO 1.

Next, we will learn how to perform TCM decoding using the Viterbi algorithm. We will also understand how to efficiently compute the free distance. We will also learn how to design TCM schemes for AWGN and Fading channels, in LO 2.

Finally, we will introduce the concept of Space Time Trellis Codes (STTC). We will also learn the STTC design criteria for fading channels, in LO 3.

8.2 The Concept of Coded Modulation

LO 1



Explain the concept of Trellis Coded Modulation (TCM) and carry out encoding using a TCM scheme.

Traditionally, coding and modulation have been considered as two separate parts of a digital communications system. The input message stream is first channel encoded (extra bits are added) and then these encoded bits are converted into an analogous waveform by the modulator. The objective of both the channel encoder and the modulator is to correct errors resulting due to the non-ideal channel. Both these blocks (the encoder and the modulator) are optimised *independently* even though their objective is the same, that is, to correct errors introduced by the channel! As we have seen, a higher

performance is possible by lowering the code rate at the cost of bandwidth expansion and increased decoding complexity. It is also possible to obtain coding gain *without* bandwidth expansion if the channel encoder is integrated with the modulator. We illustrate this by a simple example.



Example 8.1 Consider data transmission over a channel with a throughput of 2 bits/s/Hz. One possible solution is to use uncoded QPSK. Another possibility is to first use a rate 2/3 convolutional encoder (which converts 2 uncoded bits to 3 coded bits) and then use an 8-PSK signal set which has a throughput of 3 bits/s/Hz. This coded 8-PSK scheme yields the same information data throughput of the uncoded QPSK (2 bits/s/Hz). Note that both the QPSK and the 8-PSK schemes require the same bandwidth. But we know that the symbol error rate

for the 8-PSK is worse than that of QPSK for the same energy per symbol. However, the 2/3 convolutional encoder would provide some coding gain. It could be possible that the coding gain provided by the encoder outweighs the performance loss because of the 8-PSK signal set. If the coded modulation scheme performs superior to the uncoded one at the same SNR, we can claim that an improvement is achieved without sacrificing either the data rate or the bandwidth. In this example we have combined a trellis encoder with the modulator. Such a scheme is called a **Trellis Coded Modulation (TCM)** scheme.

We observe that the expansion of the signal set in order to provide redundancy results in the shrinking of the Euclidean distance between the signal points, if the average signal energy is to be kept constant (Fig. 8.1). This reduction in the Euclidean distance increases the error rate which should be compensated with coding (increase in the Hamming distance), if the coded modulation scheme is to provide an improvement. We also know that the use of a *hard-decision* demodulation prior to decoding in a coded scheme causes an irreversible loss of information. This translates to a loss of SNR. For coded modulation schemes, where the expansion of the signal set implies a power penalty, the use of *soft-decision* decoding is imperative. As a result, demodulation and decoding should be combined in a single step, and the decoder should operate on the *soft* output samples of the channel. For maximum likelihood decoding using soft-decisions, the optimal decoder chooses that code sequence which is nearest to the received sequence in terms of the Euclidean distance. Hence, an efficient coding scheme should be designed based on maximising the minimum Euclidean distance between the coded sequences rather than the Hamming distance.

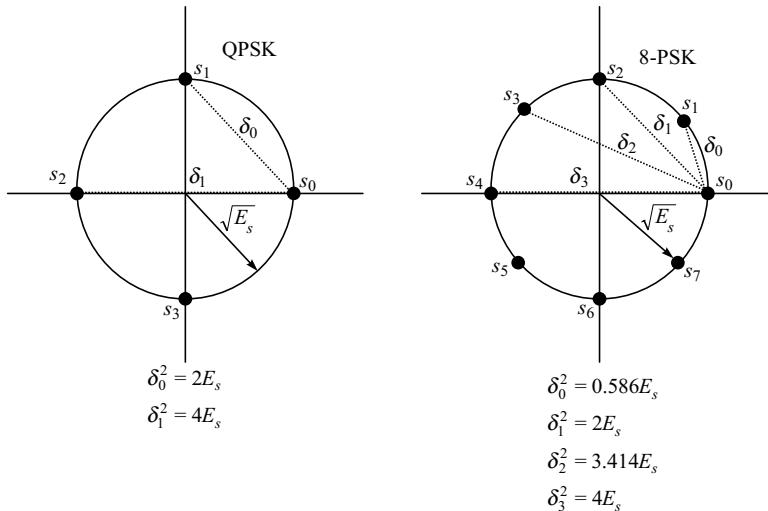


Fig. 8.1 The Euclidean distances between the signal points for QPSK and 8-PSK.

The Viterbi algorithm can be used to decode the received symbols for a TCM scheme. In the previous chapter we saw that the basic idea in Viterbi decoding is to trace out the most likely path through the trellis. The most likely path is that which is closest to the received sequence in terms of the Hamming distance. For a TCM scheme, the Viterbi decoder chooses the most likely path in terms of Euclidean distance. The performance of the decoding algorithm depends on the minimum Euclidean distance between a pair of paths forming an error event.



Definition 8.1 The minimum Euclidean distance between *any* two paths in the trellis is called the **Free Euclidean Distance**, d_{free} , of the TCM scheme.

In the previous chapter we had defined d_{free} in terms of Hamming distance between any two paths in the trellis. The minimum free distance in terms of Hamming weight could be calculated as the minimum weight of a path that deviates from the all zero path and later merges back into the all zero path at some point further down the trellis. This was a consequence of the linearity of convolutional codes. However, the same does not apply for the case of TCM, which are non-linear. It may be possible that d_{free} is the Euclidean distance between two paths in the trellis *none* of which is the all zero path. Thus, in order to calculate the free Euclidean distance for a TCM scheme, all pairs of paths have to be evaluated.

Example 8.2

Consider the convolutional encoder followed by a modulation block performing natural mapping ($000 \rightarrow s_0, 001 \rightarrow s_1, \dots, 111 \rightarrow s_7$) shown in Fig. 8.2. The rate of the encoder is $2/3$. It takes in two bits at a time (a_1, a_2) and outputs three encoded bits (c_1, c_2, c_3). The three output bits are then mapped to one of the eight possible signals in the 8-PSK signal set.

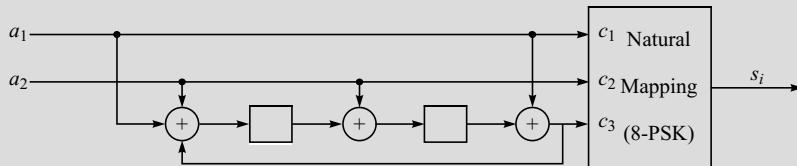


Fig. 8.2 The TCM scheme for Example 8.2.

This combined encoding and modulation can also be represented using a trellis with its branches labelled with the output symbol (s_i). The TCM scheme depicted here. This is a fully connected trellis. Each branch is labelled by a symbol from the 8-PSK constellation diagram. In order to represent the symbol allocation unambiguously, the assigned symbols to the branches are written at the front end of the trellis. The convention is as follows. Consider state 1. The branch from state 1 to state 1 is labelled with s_0 , branch from state 1 to state 2 is labelled with s_7 , branch from state 1 to state 3 is labelled with s_5 and branch from state 1 to state 4 is labelled with s_2 . So, the 4-tuple $\{s_0, s_7, s_5, s_2\}$ in front of state 1 represents the branch labels emanating from state 1 in sequence. To encode any incoming bit stream, we follow the same procedure as for convolutional encoder. However, in the case of TCM, the output is a sequence of symbols rather than a sequence of bits. Suppose we have to encode the bit stream $1\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 1\dots$, then we first group the input sequence in pairs because the input is two bits at a time. The grouped input sequence is

10 11 10 00 ...

The TCM encoder output can be obtained simply by following the path in the trellis as dictated by the input sequence. The first input pair is 10. Starting from the first node in state 0, we traverse the third branch emanating from this node as dictated by the input 10. This takes us to state 2. The symbol output for this branch is s_5 . From state 2 we move along the fourth branch as determined by the next input pair 11. The symbol output for this branch is s_1 . In this manner, the output symbols corresponding to the given input sequence is

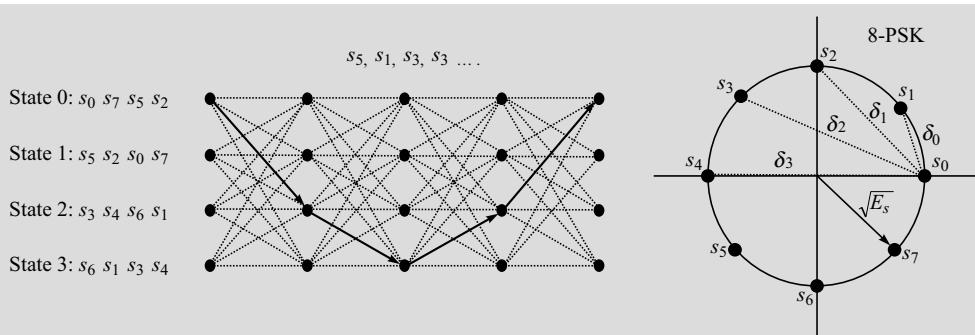


Fig. 8.3 The path in the trellis corresponding to the input sequence 10 11 10 00 ...



The path in the trellis is depicted by the bold lines in Fig. 8.3. As in the case of convolutional encoder, in TCM too, every encoded sequence corresponds to a unique path in the trellis. The objective of the decoder is to recover this path from the trellis diagram.

Example 8.3 Consider the TCM scheme shown in Example 8.2. The free Euclidean distance, d_{free} of the TCM scheme can be found by inspecting all possible pairs of paths in the trellis. The two paths that are separated by the minimum squared Euclidean distance (which yields the d_{free}) are shown with bold lines in the trellis diagram given in Fig. 8.4.

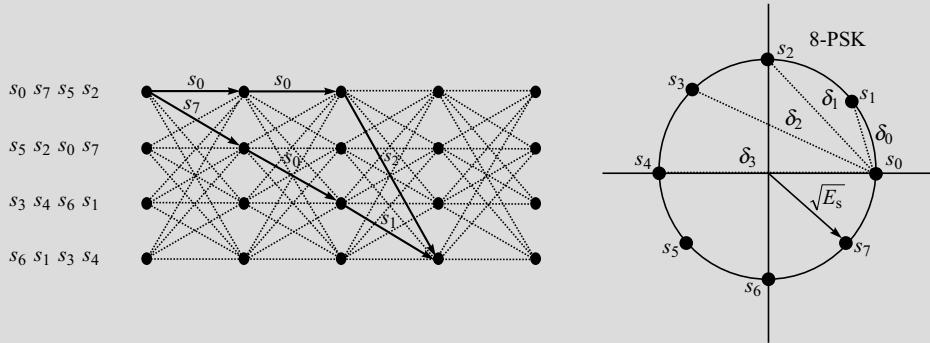


Fig. 8.4 The two paths in the trellis that have the squared free Euclidean distance, d_{free}^2 .

$$\begin{aligned} d_{free}^2 &= d_E^2(s_0, s_7) + d_E^2(s_0, s_0) + d_E^2(s_2, s_1) \\ &= \delta_0^2 + 0 + \delta_0^2 = 2\delta_0^2 = 1.172 E_s \end{aligned}$$

It can be seen that in this case, the error event that results in d_{free} does not involve the all zero sequence. As mentioned before, in order to find the d_{free} , we must evaluate *all* possible pairs of paths in the trellis. It is not sufficient just to evaluate the paths diverging from and later merging back into the all zero paths because of the non-linear nature of TCM.

We must now develop a method to compare the coded scheme with the uncoded one. We introduce the concept of a coding gain in the following example.

Definition 8.2 The difference between the values of the SNR for the coded and uncoded schemes required to achieve the same error probability is defined as the **Coding Gain**, g .

$$g = \text{SNR}_{\text{uncoded}} - \text{SNR}_{\text{coded}} \quad (8.1)$$

At high SNR, the coding gain can be expressed as



$$g_{\infty} = g|_{\text{SNR} \rightarrow \infty} = 10 \log \frac{(d_{\text{free}}^2 / E_s)_{\text{coded}}}{(d_{\text{free}}^2 / E_s)_{\text{uncoded}}} \quad (8.2)$$

where g_{∞} represents the **Asymptotic Coding Gain** and E_s is the average signal energy. For uncoded schemes, d_{free} is simply the minimum Euclidean distance between the signal points.

Example 8.4 Consider the TCM scheme discussed in Example 8.2 in which the input takes in 2 bits at a time. If we were to send uncoded bits, we would employ QPSK. The d_{free}^2 for the uncoded scheme (QPSK) is $2E_s$ from Fig. 8.1. From Example 8.3 we have $d_{\text{free}}^2 = 1.172E_s$ for our TCM scheme. The asymptotic coding gain is then given by

$$g_{\infty} = 10 \log \frac{1.172}{2} = -2.3 \text{ dB.}$$

This implies that the performance of our TCM scheme is actually *worse* than the uncoded scheme. A quick look at the convolutional encoder used in this example suggests that it has good properties in terms of Hamming distance. In fact, it can be verified that this convolutional encoder is optimal in the sense of maximising the free Hamming distance. However, the encoder fails to perform well for the case of TCM. This illustrates the point that TCM schemes must be designed to maximise the Euclidean distance rather than the Hamming distance.

For a fully connected trellis, as discussed in Example 8.2, by a proper choice of the mapping scheme we can improve the performance. In order to design a better TCM scheme, it is possible to directly work from the trellis onwards. The objective is to assign the 8 symbols from the 8-PSK signal set in such a manner that the d_{free} is maximised. One approach is to use an exhaustive computer search. There are a total of 16 branches that have to be assigned labels (symbols) from time t_k to t_{k+1} . We have 8 symbols to choose from. Thus, an exhaustive search would involve 8^{16} different cases!

Another approach is to assign the symbols to the branches in the trellis in a heuristic manner so as to increase the d_{free} . We know that an error event consists of a path diverging in one state and then merging back after one or more transitions, as depicted in Fig. 8.5. The Euclidean distance associated with such an error events can be expressed as

$$d_{\text{total}}^2 = d_E^2(\text{diverging pair of paths}) + \dots + d_E^2(\text{re-merging pair of paths}) \quad (8.3)$$

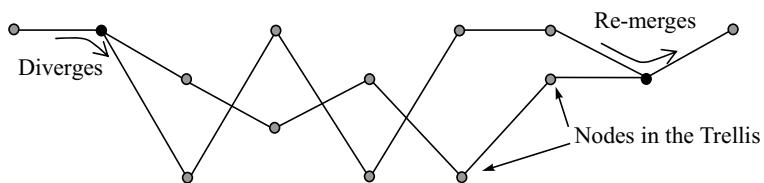


Fig. 8.5 An error event.

Thus, in order to design a TCM scheme with a large d_{free}^2 , we can at least ensure that the d_E^2 (diverging pair of paths) and the d_E^2 (re-merging pair of paths) are as large as possible. In TCM schemes, a redundant 2^{m+1} -ary signal set is often used to transmit m bits in each signalling interval. The m input bits are first encoded by a $m/(m + 1)$ convolutional encoder. The resulting $m + 1$ output bits are then mapped to the signal points of the 2^{m+1} -ary signal set. The mapping is done in such a manner so as to maximise the minimum Euclidean distance between the different paths in the trellis. This is done using a rule called **mapping by Set Partitioning**.

8.3 Mapping by Set Partitioning

LO 1

The mapping by set partitioning is based on successive partitioning of the expanded 2^{m+1} -ary signal set into subsets with increasing minimum Euclidean distances. Each time we partition the set, we reduce the number of the signal points in the subset, but increase the minimum distance between the signal points in the subset. The set partitioning can be understood by the following example.

Example 8.5 Consider the set partitioning of 8-PSK. Before partitioning, the minimum Euclidean distance of the signal set is $\Delta_0 = \delta_0$. In the first step, the 8 points in the constellation diagram are subdivided into two subsets, A_0 and A_1 , each containing 4 signal points as shown in Fig. 8.6. As a result of this first step, the minimum Euclidean distance of each of the subsets is now $\Delta_1 = \delta_1$, which

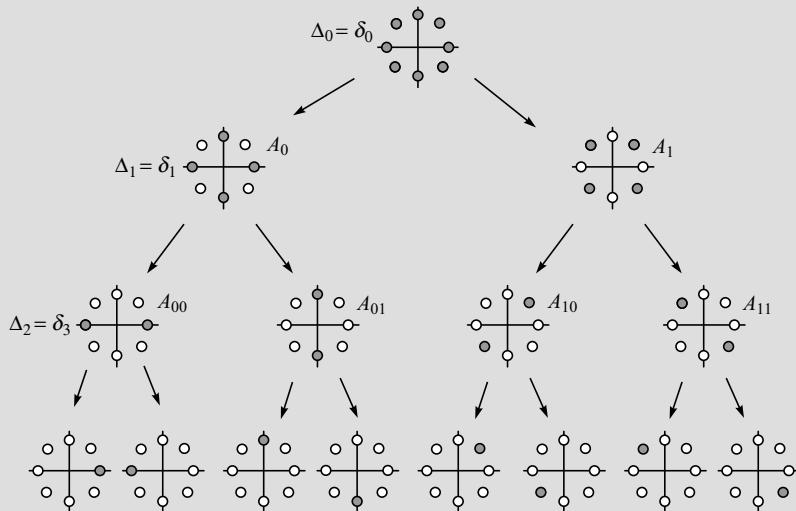


Fig. 8.6 Set partitioning of the 8-PSK signal set.

is larger than the minimum Euclidean distance of the original 8-PSK. We continue this procedure and subdivide the sets A_0 and A_1 and into two subsets each, $A_0 \rightarrow \{A_{00}, A_{01}\}$ and $A_1 \rightarrow \{A_{10}, A_{11}\}$. As a result of this second step, the minimum Euclidean distance of each of the subsets is now $\Delta_2 = \delta_3$. Further subdivision results in one signal point per subset.

Consider the expanded 2^{m+1} -ary signal set used for TCM. In general it is not necessary to continue the process of set partitioning until the last stage. The partitioning can be stopped as soon as the minimum distance of a subset is larger than the desired minimum Euclidean distance of the TCM scheme to be designed. Suppose the desired Euclidean distance is obtained just after the $\tilde{m} + 1^{\text{th}}$ set partitioning step ($\tilde{m} \leq m$). It can be seen that after $\tilde{m} + 1$ steps we have $2^{\tilde{m}+1}$ subsets and each subset contains $2^{m-\tilde{m}}$ signal points.

A general structure of a TCM encoder is given in Fig. 8.7. It consists of m input bits of which the \tilde{m} bits are fed into a rate $\tilde{m}/(\tilde{m}+1)$ convolutional encoder while the remaining $m - \tilde{m}$ bits are left uncoded. The $\tilde{m} + 1$ output bits of the encoder along with the $m - \tilde{m}$ uncoded bits are then input to the signal mapper. The signal mapper uses the $\tilde{m} + 1$ bits from the convolutional encoder to select one of the possible $2^{\tilde{m}+1}$ subsets. The remaining $m - \tilde{m}$ uncoded bits are used to select one of the $2^{m-\tilde{m}}$ signals from this subset. Thus, the input to the TCM encoder is m bits and the output is a signal point chosen from the original constellation.

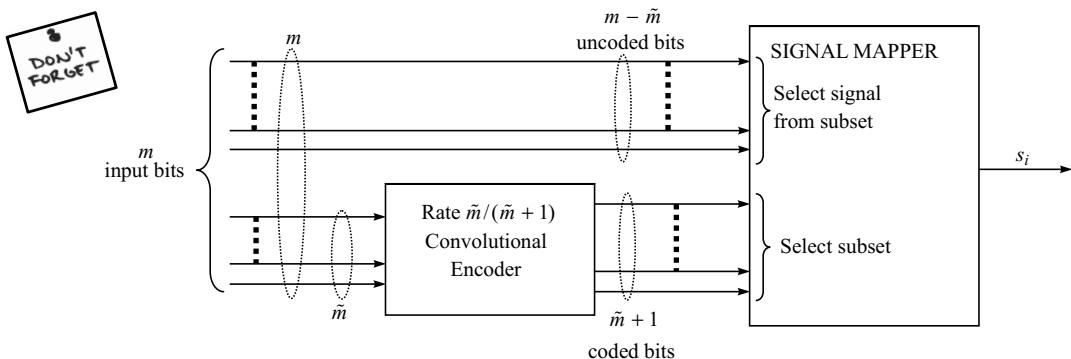


Fig. 8.7 The general structure of a TCM encoder that processes m input bits.

For the TCM encoder shown in Fig. 8.7, we observe that $m - \tilde{m}$ uncoded bits have no effect on the state of the convolutional encoder because the input to the convolutional encoder is not being altered. Thus, we can change the first $m - \tilde{m}$ bits of the total m input bits *without* changing the encoder state. This implies that $2^{m-\tilde{m}}$ **parallel transitions** exist between states. These parallel transitions are associated with the signals of the subsets in the lowest layer of the set partitioning tree. For the case of $m = \tilde{m}$, the states are joined by single transitions.

Let us denote the minimum Euclidean distance between parallel transitions by $\Delta_{\tilde{m}+1}$ and the minimum Euclidean distance between non-parallel paths of the trellis by $d_{\text{free}}(\tilde{m})$. The free Euclidean distance of the TCM encoder shown in Fig. 8.7 can then be written as

$$d_{\text{free}} = \min [\Delta_{\tilde{m}+1}, d_{\text{free}}(\tilde{m})] \quad (8.4)$$

Example 8.6 Consider the TCM scheme proposed by Ungerboeck. It is designed to maximise the free Euclidean distance between coded sequences. It consists of a rate 2/3 convolutional encoder coupled with an 8-PSK signal set mapping. The encoder is given in Fig. 8.8 and the corresponding trellis diagram in Fig. 8.9.

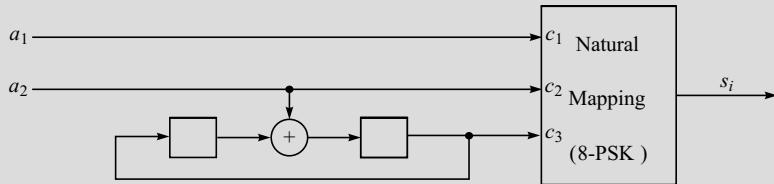


Fig. 8.8 The TCM encoder for Example 8.6.

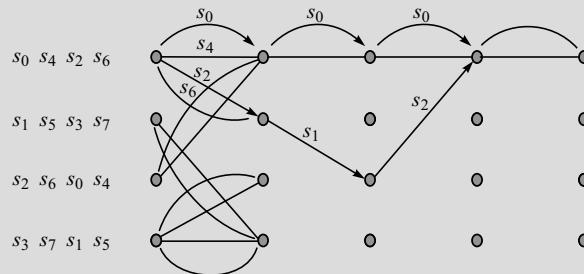


Fig. 8.9 The trellis diagram for the encoder in Example 8.6.

For this encoder $m = 2$ and $\tilde{m} = 1$, which implies that there are $2^{m-\tilde{m}} = 2^1 = 2$ parallel transitions between each state. The minimum squared Euclidean distance between parallel transitions is

$$\Delta_{\tilde{m}+1}^2 = \Delta_2^2 = \delta_3^2 = 4E_s$$

The minimum squared Euclidean distance between non-parallel paths in the trellis, $d_{free}(\tilde{m})$, is given by the error event shown in Fig. 8.9 by bold lines. From Fig. 8.9, we have

$$\begin{aligned} d_{free}^2(\tilde{m}) &= d_E^2(s_0, s_2) + d_E^2(s_0, s_1) + d_E^2(s_0, s_4) \\ &= \delta_1^2 + \delta_0^2 + \delta_1^2 = 4.586 E_s. \end{aligned}$$

The error events associated with the parallel paths have the minimum squared Euclidean distance among all the possible error events. Therefore, the minimum squared Euclidean distance for the TCM scheme is $d_{free}^2 = \min[\Delta_{\tilde{m}+1}^2, d_{free}^2(\tilde{m})] = 4E_s$. The asymptotic coding gain of this scheme is

$$g_\infty = 10 \log \frac{4}{2} = 3 \text{ dB}$$

DID YOU KNOW ? This shows that the TCM scheme proposed by Ungerboeck shows an improvement of 3 dB over the uncoded QPSK. This example illustrates the point that the combined coded modulation scheme can compensate for the loss from the expansion of the signal set by the coding gain achieved by the convolutional encoder. Further, for the non-parallel paths

$$\begin{aligned} d_{\text{total}}^2 &= d_E^2(\text{diverging pair of paths}) + \dots + d_E^2(\text{re-merging pair of paths}) \\ &= \delta_1^2 + \dots + \delta_1^2 = (\delta_1^2 + \delta_1^2) + \dots = \delta_3^2 + \dots = 4E_s + \dots \end{aligned}$$

However, the minimum squared Euclidean distance for the parallel transition is $\delta_3^2 = 4E_s$. Hence, the minimum squared Euclidean distance of this TCM scheme is determined by the parallel transitions.

8.4 Ungerboeck's TCM Design Rules

LO1

In 1982, Ungerboeck proposed a set of design rules for maximising the free Euclidean distance for TCM schemes. These design rules are based on heuristics.

Rule 1: Parallel transitions, if present, must be associated with the signals of the subsets in the lowest layer of the set partitioning tree. These signals have the minimum Euclidean distance $\Delta_{\tilde{m}+1}$.

Rule 2: The transitions originating from or merging into one state must be associated with signals of the first step of set partitioning. The Euclidean distance between these signals is at least Δ_1 .

Rule 3: All signals are used with equal frequency in the trellis diagram.

Example 8.7 We next wish to improve upon the TCM scheme proposed in Example 8.6. We observed in Example 8.6 that the parallel transitions limit the d_{free}^2 . Therefore, we must come up with a trellis that has no parallel transitions. The absence of parallel paths would imply that the d_{free}^2 is not limited to δ_3^2 , the maximum possible separation between two signal points in the 8-PSK constellation diagram. Consider the trellis diagram shown in Fig. 8.10. The trellis has 8 states. There are no parallel transitions in the trellis diagram. We wish to assign the symbols from an 8-PSK signal set to the branches of this trellis according to the Ungerboeck rules.

Since there are no parallel transitions here, we start directly with Ungerboeck's second rule. We must assign the transitions originating from or merging into one state with signals from the first step of set partitioning. We will refer to Fig. 8.6 for the set partitioning diagram for 8-PSK. The first step of set partitioning yields two subsets, A_0 and A_1 , each consists of four signal points. We first focus on the diverging paths. Consider the topmost node (state S_0). We assign to these four diverging paths the signals s_0, s_4, s_2 and s_6 . Note that they all belong to the subset A_0 . For the next node (state S_1), we assign the signals s_1, s_5, s_3 and s_7 belonging to the subset A_1 . For the next node (state S_2), we assign the signals s_4, s_0, s_6 and s_2 belonging to the subset A_0 . The order has been shuffled to ensure that at the re-merging end we still have with signals from the first step of set partitioning. If we observe the four paths that merge into the node of state S_0 , their branches are labelled s_0, s_4, s_2 and s_6 , which belong to A_0 . This clever assignment has ensured that the transitions originating from or merging into one state are labelled with signals from the first step of set partitioning, thus satisfying rule 2. It can be verified that all the signals have been used with equal frequency. We did not have to make any special effort to ensure this point.

The error event corresponding to the squared free Euclidean distance is shown in the trellis diagram with bold lines. The squared free Euclidean distance of this TCM scheme is

$$\begin{aligned} d_{\text{free}}^2 &= d_E^2(s_0, s_6) + d_E^2(s_0, s_7) + d_E^2(s_0, s_6) \\ &= \delta_1^2 + \delta_0^2 + \delta_1^2 = 4.586 E_s \end{aligned}$$

In comparison to uncoded QPSK, this translates to an asymptotic coding gain of

$$g_{\infty} = 10 \log \frac{4.586}{2} = 3.60 \text{ dB}$$

Thus, at the cost of added encoding and decoding complexity, we have achieved a 0.6 dB gain over the TCM scheme discussed in Example 8.6.

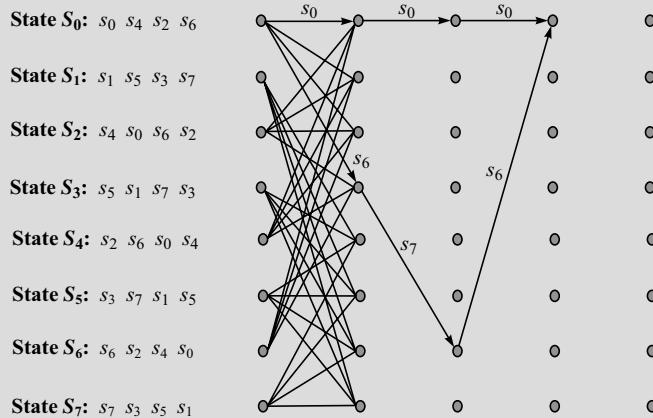


Fig. 8.10 The trellis diagram for the encoder in Example 8.7.

Example 8.8 Consider the 8-state, 8-PSK TCM scheme discussed in Example 8.7. The equivalent systematic encoder realisation with feedback is given in Fig. 8.11.

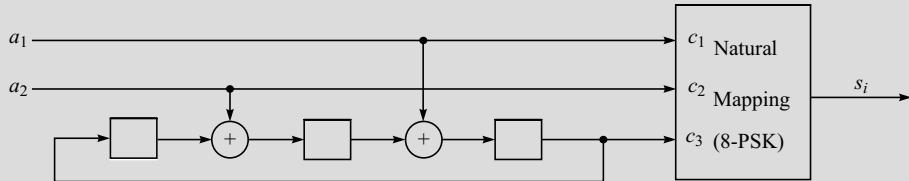


Fig. 8.11 The TCM encoder for Example 8.8.

Let us represent the output of the convolutional encoder shown in Fig. 8.11 in terms of the input and the delayed versions of the input. From Fig. 8.11, we have

$$c_1(D) = a_1(D)$$

$$c_2(D) = a_2(D)$$

$$c_3(D) = \left(\frac{D^2}{1+D^3} \right) a_1(D) + \left(\frac{D}{1+D^3} \right) a_2(D)$$

Therefore, the generator polynomial matrix of this encoder is

$$\mathbf{G}(D) = \begin{bmatrix} 1 & 0 & \frac{D^2}{1+D^3} \\ 0 & 1 & \frac{D}{1+D^3} \end{bmatrix}$$

and the parity check polynomial matrix, $\mathbf{H}(D)$, satisfying $\mathbf{G}(D) \cdot \mathbf{H}^T(D) = \mathbf{0}$ is

$$\mathbf{H}(D) = [D^2 \ D \ 1 + D^3]$$

We can re-write the parity check polynomial matrix $\mathbf{H}(D) = [H_1(D) H_2(D) H_3(D)]$, where



$$H_1(D) = D^2 = (000\ 100)_{\text{binary}} = (04)_{\text{octal}}$$

$$H_2(D) = D = (000\ 010)_{\text{binary}} = (02)_{\text{octal}}$$

$$H_3(D) = 1 + D^3 = (001\ 001)_{\text{binary}} = (11)_{\text{octal}}$$

Table 8.1 gives the encoder realisation and asymptotic coding gains of some of the good TCM codes constructed for the 8-PSK signal constellation. Almost all of these TCM schemes have been found by exhaustive computer searches. The coding gain is given with respect to the uncoded QPSK. The parity check polynomials are expressed in the octal form.

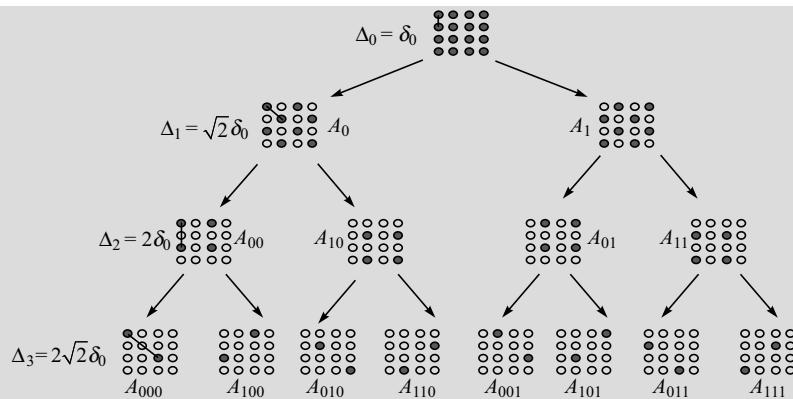
Table 8.1 TCM schemes using 8-PSK

No. of states	H_1	H_2	H_3	d_{free}^2/E_s	$g_\infty(\text{dB})$
4	-	2	5	4.00	3.01
8	04	02	11	4.58	3.60
16	16	04	23	5.17	4.13
32	34	16	45	5.75	4.59
64	066	030	103	6.34	5.01
128	122	054	277	6.58	5.17
256	130	072	435	8.51	5.75

Example 8.9 We now look at a TCM scheme that involves 16QAM. The TCM encoder takes in 3 bits and outputs one symbol from the 16QAM constellation diagram. This TCM scheme has a throughput of 3 bits/s/Hz and we will compare it with uncoded 8-PSK, which also has a throughput of 3 bits/s/Hz.

Let the minimum distance between two points in the signal constellation of 16QAM be δ_0 as depicted in Fig. 8.12. It is assumed that all the signals are equi-probable. Then the average signal energy of a 16QAM signal is obtained as

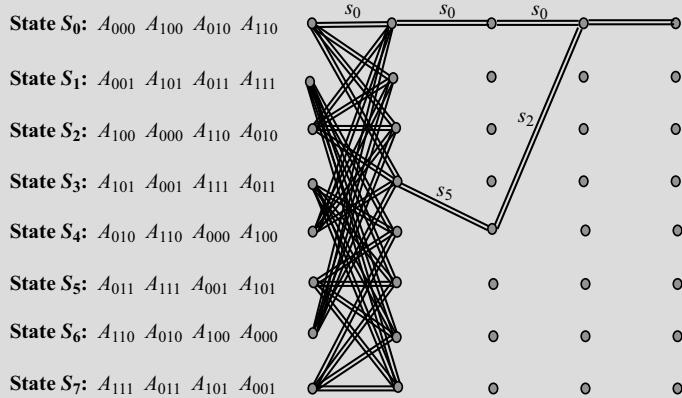
$$E_s = \frac{1}{16} (2\delta_0^2 + 10\delta_0^2 + 10\delta_0^2 + 18\delta_0^2) = \frac{10}{4}\delta_0^2$$

**Fig. 8.12** Set partitioning of 16QAM.

Thus we have,

$$\delta_0 = 2\sqrt{\frac{E_s}{10}}$$

The trellis diagram for the 16QAM TCM scheme is given in Fig. 8.13. The trellis has 8 states. Each node has 8 branches emanating from it because the encoder takes in 3 input bits at a time ($2^3 = 8$). The encoder realisation is given in Fig. 8.14. The Ungerboeck design rules are followed to assign the symbols to the different branches. The branches diverging from a node and the branches merging back to a node are assigned symbols from the set A_0 and A_1 . The parallel paths are assigned symbols from the lowest layer of the set partitioning tree (A_{000}, A_{001} , etc.).

**Fig. 8.13** Trellis diagram for the 16QAM TCM scheme.

The squared Euclidean distance between any two parallel paths is $\Delta_3^2 = 8\delta_0^2$. This is by design as we have assigned symbols to the parallel paths from the lowest layer of the set partitioning tree. The minimum squared Euclidean distance between non-parallel paths is

$$d_E^2 = \Delta_1^2 + \Delta_0^2 + \Delta_1^2 = 5\delta_0^2.$$

Therefore, the free Euclidean distance for the TCM scheme is

$$d_{free}^2 = \min[8\delta_0^2, 5\delta_0^2] = 5\delta_0^2 = 2E_s.$$

Note that the free Euclidean distance is determined by the non-parallel paths rather than the parallel paths. We now compare the TCM scheme with the uncoded 8-PSK, which has the same throughput. For uncoded 8-PSK, the minimum squared Euclidean distance is $(2 - \sqrt{2}) E_s$. Thus, the asymptotic coding gain for this TCM encoder is

$$g_{\infty} = 10 \log \frac{2}{2 - \sqrt{2}} = 5.3 \text{ dB}$$

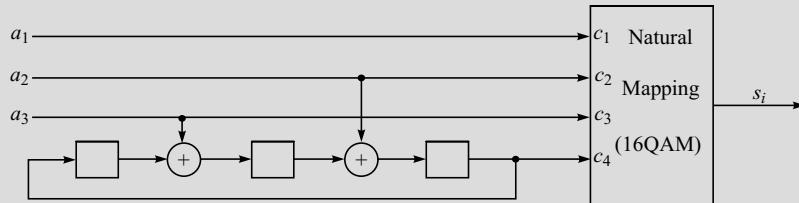


Fig. 8.14 The equivalent systematic encoder for the trellis diagram shown in Fig. 8.13.

INDUSTRIAL RELEVANCE



Example 8.10 The TCM scheme for the V.34 modem standard is shown in Fig. 8.15. It uses a 16-state trellis and provides a coding gain of 4.66 dB. A 192-point, 2-dimensional signal constellation is used, where the symbols are denoted by 0, 1, 2, 3. The data rate achieved by this modem is 33.6 kb/s on most standard telephone lines.

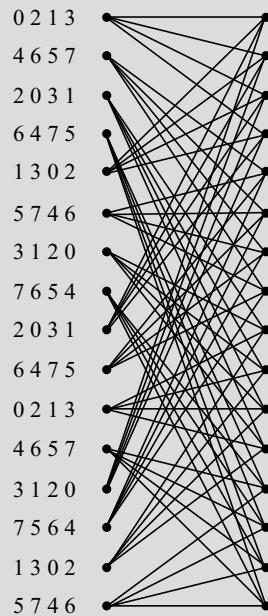


Fig. 8.15 The TCM scheme used in V.34 modem standard.



INDUSTRIAL RELEVANCE

In the late 1980s and early 1990s, modem was an exciting means of communication over telephone lines. Modem companies, at that time, started with a data rate of 9.6 kbps by employing QAM at baud-rate equal to 2,400 symbols/s. However, Shannon's theorem predicted a much higher data rate (~ 33 kbps) for telephone lines, assuming a bandwidth of 4 kHz and an SNR of 25 dB. Recall that SNR in dB is calculated as $10\log_{10}(\text{SNR})$. Hence, $25 \text{ dB SNR} = 10^{2.5} = 316.22$. For these parameters, the capacity of the telephone channel is $C = W \log_2(1 + \text{SNR}) = 4 \times 10^3 \log_2(1 + 316.22) = 33.238$ b/s. However, 9.6 kbps was much below the channel capacity. In the mid-1990s, the use of TCM schemes permitted engineers to increase the modem data rates progressively: 14.4 kbps, 28.8 kbps and finally 33.6 kbps. The various analogue modem standards: V.32 (19.2 kbps), V.34 (28.8 kbps) and V.34+ (33.6 kbps) all used TCM schemes. Subsequently, the V.90 modem standard (hello digital, bye-bye analogue!) used PCM + TCM, and achieved a download data rate close to 56 kbps. In the late 1990s, V.92 standard came into being where PCM was used for both upstream (48 kbps) and downstream (56 kbps). At the turn of the century, the modem dream-run was cut-short by the arrival of broadband access – both wireline broadband access and wireless broadband access.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 1:

1. Draw the constellation diagram for QPSK, 16QAM, 32 Cross, 64QAM and 128 Cross. Indicate the minimum symbol separation by δ_0 . S
2. Consider the TCM scheme shown in Fig. 8.3. Encode the following input bit stream: 1 1 1 0 0 1 0 0 0 ... Also, give the state transitions. S
3. Consider the TCM scheme shown in Fig. 8.16. Find the d_{free}^2 and the asymptotic coding gain. S

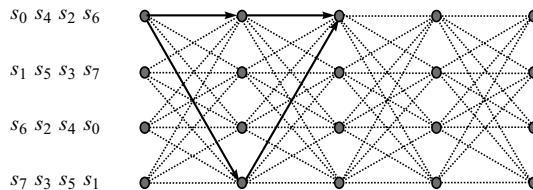
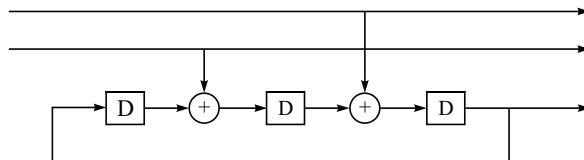


Fig. 8.16

Levels of Difficulty

- S **Simple:** Level 1 and Level 2 Category
- M **Medium:** Level 3 and Level 4 Category
- D **Difficult:** Level 5 and Level 6 Category

4. Consider the rate 2/3 convolutional encoder coupled with a signal mapper. Draw the trellis diagram for the encoder and label the branches as per Ungerboeck's design rules. Find the asymptotic coding gain for this TCM scheme.



5. Carry out set-partitioning on 8-ASK.

If you have successfully solved the above problems,
you have mastered LO 1. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/617>



8.5 TCM Decoder

LO 2



Know how to perform TCM decoding and design TCM schemes for AWGN and Fading channels.

We have seen that, like convolutional codes, TCM schemes are also described using trellis diagrams. Any input sequence to a TCM encoder gets encoded into a sequence of symbols based on the trellis diagram. The encoded sequence corresponds to a particular path in this trellis diagram. There exists a one-to-one correspondence between an encoded sequence and a path within the trellis. The task of the TCM decoder is simply to identify the most likely path in the trellis. This is based on the maximum likelihood criterion. As seen in the previous chapter, an efficient search method is to use the Viterbi algorithm.

For soft decision decoding of the received sequences using the Viterbi algorithm, each trellis branch is labelled by the branch metric based on the observed received sequence. Using the maximum likelihood decoder for the Additive White Gaussian Noise (AWGN) channels, the branch metric is defined as the Euclidean distance between the coded sequence and the received sequence (see Section 7.7). The Viterbi decoder finds a path through the trellis which is closest to the received sequence in the Euclidean distance sense.



Definition 8.3 The **Branch Metric** for a TCM scheme designed for AWGN channel is the *Euclidean distance* between the received signal and the signal associated with the corresponding branch in the trellis.

LO 2

8.6 Performance Evaluation for AWGN Channel

There are different performance measures for a TCM scheme designed for an AWGN channel. We have already discussed the asymptotic coding gain, which is based on free Euclidean distance, d_{free} . We will now look at some other parameters that are used to characterise a TCM code.

Definition 8.4 The Average Number of Nearest Neighbours at free distance, $N(d_{\text{free}})$, gives the average number of paths in the trellis with free Euclidean distance d_{free} from a transmitted sequence. This number is used in conjunction with d_{free} for the evaluation of the error event probability.

Definition 8.5 Two finite length paths in the trellis form an **Error Event** if they start from the same state, diverge and then later merge back. An error event of length l is defined by two coded sequences s_n and \hat{s}_n ,



such that

$$\begin{aligned} s_n &= (s_n, s_{n+1}, \dots, s_{n+l+1}) \\ \hat{s}_n &= (\hat{s}_n, \hat{s}_{n+1}, \dots, \hat{s}_{n+l+1}), \\ s_i &\neq \hat{s}_i, i = n + 1, \dots, n + l \end{aligned} \tag{8.5}$$

Definition 8.6 The probability of an error event starting at time n , given that the decoder has estimated the correct transmitter state at that time is called the **Error Event Probability**, P_e .

The performance of TCM schemes is generally evaluated by means of upper bounds on error event probability. It is based on the generating function approach. Let us consider again the Ungerboeck model for rate $m/(m+1)$ TCM scheme as shown in Fig. 8.7. The encoder takes in m bits at a time and encodes it to $m+1$ bits, which are then mapped by a memoryless mapper, $f(\cdot)$, on to a symbol s_i . Let us call the binary $(m+1)$ -tuples c_i as the labels of the signals s_i . We observe that there is a one-to-one correspondence between a symbol and its label. Hence, an error event of length l can be equivalently described by two sequences of labels

$$\mathbf{C}_l = (c_k, c_{k+1}, \dots, c_{k+l-1}) \text{ and } \mathbf{C}'_l = (c'_k, c'_{k+1}, \dots, c'_{k+l-1}) \tag{8.6}$$

where, $c'_k = c_k \oplus e_k, c'_{k+1} = c_{k+1} \oplus e_{k+1}, \dots$, and $\mathbf{E}_l = (e_k, e_{k+1}, \dots, e_{k+l-1})$ is a sequence of binary error vectors. The mathematical symbol \oplus represents binary (modulo-2) addition. An error event of length l occurs when the decoder chooses, instead of the transmitted sequence \mathbf{C}_l , the sequence \mathbf{C}'_l which corresponds to a path in the trellis diagram that diverges from the original transmitted path and re-merges back exactly after l time intervals. To find the probability of error we need to sum over all possible values of l and the probabilities of error events of length l (i.e., joint probabilities that \mathbf{C}_l is transmitted and \mathbf{C}'_l is detected). The upper bound on the probability of error is obtained by the following union bound

$$P_e \leq \sum_{l=1}^{\infty} \sum_{s_l} \sum_{s'_l \neq s_l} P(s_l) P(s_l, s'_l) \tag{8.7}$$

where $P(s_l, s'_l)$ denotes the pairwise error probability (i.e., probability that the sequence s_l is transmitted and the sequence s'_l is detected). Assuming a one-to-one correspondence between a symbol and its label, we can write

$$\begin{aligned} P_e &\leq \sum_{l=1}^{\infty} \sum_{\mathbf{C}_l} \sum_{\mathbf{C}'_l \neq \mathbf{C}_l} P(\mathbf{C}_l) P(\mathbf{C}_l, \mathbf{C}'_l) \\ &= \sum_{l=1}^{\infty} \sum_{\mathbf{C}_l} \sum_{\mathbf{E}_l \neq 0} P(\mathbf{C}_l) P(\mathbf{C}_l, \mathbf{C}_l \oplus \mathbf{E}_l) \end{aligned} \quad (8.8)$$

The pairwise error probability $P_2(\mathbf{C}_l, \mathbf{C}_l \oplus \mathbf{E}_l)$ can be upper-bounded by the Bhattacharyya bound (see Problem 8.13) as follows

$$\begin{aligned} P_2(\mathbf{C}_l, \mathbf{C}_l \oplus \mathbf{E}_l) &\leq e^{-\left\{ \frac{1}{4N_0} \|f(\mathbf{C}_l) - f(\mathbf{C}'_l)\|^2 \right\}} \\ &= e^{-\left\{ \frac{1}{4N_0} \sum_{n=1}^L \|f(c_l) - f(c'_l)\|^2 \right\}} \end{aligned} \quad (8.9)$$

where $\|\cdot\|^2$ represents the squared Euclidean distance and $f(\cdot)$ is the memoryless mapper. Let $D = e^{-\left\{ \frac{1}{4N_0} \sum_{n=1}^L \|f(c_l) - f(c'_l)\|^2 \right\}}$ (for additive white Gaussian noise channel with single sided power spectral density N_0), then

$$P_2(\mathbf{C}_l, \mathbf{C}_l \oplus \mathbf{E}_l) \leq D^{\|f(\mathbf{C}_l) - f(\mathbf{C}'_l)\|^2} = D^{d_E^2(f(\mathbf{C}_l), f(\mathbf{C}'_l))} \quad (8.10)$$

where $d_E^2(f(\mathbf{C}_l), f(\mathbf{C}'_l))$ represents the squared Euclidean distance between the symbol sequences s_l and s'_l . Next, define the function

$$W(\mathbf{E}_l) = \sum_{\mathbf{C}_l} P(\mathbf{C}_l) D^{\|f(\mathbf{C}_l) - f(\mathbf{C}_l \oplus \mathbf{E}_l)\|^2} \quad (8.11)$$

We can now write the probability of error as

$$P_e \leq \sum_{l=1}^{\infty} \sum_{\mathbf{E}_l \neq 0} W(\mathbf{E}_l) \quad (8.12)$$

From the above equation, we observe that the probability of error is upper-bounded by a sum over all possible error events, \mathbf{E}_l . Note that

$$d_E^2(f(\mathbf{C}_l), f(\mathbf{C}_l \oplus \mathbf{E}_l)) = \sum_{i=1}^L d_E^2(f(c_i), f(c_i \oplus e_i)) \quad (8.13)$$

We now introduce the concept of an error state diagram which is essentially a graph whose branches have matrix labels. We assume that the source symbols are equally probable with probabilities $2^{-m} = 1/M$.

Definition 8.7 The **Error Weight Matrix**, $\mathbf{G}(e_i)$ is an $N \times N$ matrix whose element in the p^{th} row and q^{th} column is defined as

 $[\mathbf{G}(e_i)]_{p,q} = \frac{1}{M} \sum_{c_{p \rightarrow q}} D^{\|f(c_{p \rightarrow q}) - f(c_{p \rightarrow q} \oplus e_i)\|^2}$, if there is a transition from state p to state q , and

$$[\mathbf{G}(e_i)]_{p,q} = 0, \quad \text{if there is no transition from state } p \text{ to state } q \text{ of the trellis}, \quad (8.14)$$

where $c_{p \rightarrow q}$ are the label vectors generated by the transition from state p to state q .

The summation accounts for the possible parallel transitions (parallel paths) between states in the trellis diagram. The entry (p, q) in the matrix \mathbf{G} provides an upperbound on the probability that an error event occurs starting from the node p and ending on q . Similarly, $(1/N)\mathbf{G}\mathbf{1}$ is a vector whose p^{th} entry is a bound on the probability of any error event starting from node p . Now, to any sequence $\mathbf{E}_l = \mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_l$, there corresponds a sequence of l error weight matrices $\mathbf{G}(\mathbf{e}_1), \mathbf{G}(\mathbf{e}_1), \dots, \mathbf{G}(\mathbf{e}_l)$. Thus we have

$$W(\mathbf{E}_l) = \frac{1}{N} \mathbf{1}^T \prod_{n=1}^l \mathbf{G}(\mathbf{e}_n) \mathbf{1} \quad (8.15)$$

where $\mathbf{1}$ is a column N -vector, all elements of which are unity. We make the following observations:

- (i) For any matrix \mathbf{A} , $\mathbf{1}^T \mathbf{A} \mathbf{1}$ represents the sum of all entries of \mathbf{A} .
- (ii) The element (p, q) of the matrix $\prod_{n=1}^l \mathbf{G}(\mathbf{e}_n)$ enumerate the Euclidean distance involved in transition from state p to state q in exactly l steps.

Our next job is to relate the above analysis to the probability of error, P_e . It should be noted that the error vectors $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_l$ are not independent. The error state diagram has a structure determined only by the linear convolutional code and differs from the code state diagram only in the denomination of its state and branch labels ($\mathbf{G}(\mathbf{e}_i)$). Since the error vectors \mathbf{e}_i are simply the differences of the vectors \mathbf{c}_i , the connections among the vectors \mathbf{e}_i are the same as that among the vectors \mathbf{c}_i . Therefore, from (8.12) and (8.15) we have

$$P_e \leq T(D)|_{D=e^{-1/4N_0}} \quad (8.16)$$

where

$$T(D) = \frac{1}{N} \mathbf{1}^T \mathbf{G} \mathbf{1} \quad (8.17)$$

and the matrix

$$\mathbf{G} = \sum_{l=1}^{\infty} \sum_{\mathbf{E}_l \neq 0} \prod_{n=1}^l \mathbf{G}(\mathbf{e}_n) \quad (8.18)$$

is the matrix transfer function of the error state diagram. $T(D)$ is called the **Scalar Transfer Function** or simply the **Transfer Function** of the error state diagram.

Example 8.11 Consider a rate $1/2$ TCM scheme with $m = 1$, and $M = 4$. It takes one bit at a time and encodes it into two bits, which are then mapped to one of the four QPSK symbols. The two-state trellis diagram and the symbol allocation from the 4-PSK constellation are given in Fig. 8.17.

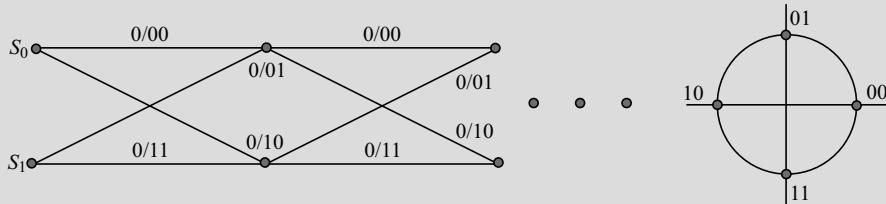


Fig. 8.17 Two-state trellis diagram.

Let us denote the error vector by $\mathbf{e} = (e_2 \ e_1)$. Then, from (8.14)

$$\begin{aligned}\mathbf{G}(e_2 e_1) &= \frac{1}{2} \begin{bmatrix} D \|f(00) - f(00 \oplus e_2 e_1)\|^2 & D \|f(10) - f(10 \oplus e_2 e_1)\|^2 \\ D \|f(01) - f(01 \oplus e_2 e_1)\|^2 & D \|f(11) - f(11 \oplus e_2 e_1)\|^2 \end{bmatrix} \\ &= \frac{1}{2} \begin{bmatrix} D \|f(00) - f(e_2 e_1)\|^2 & D \|f(10) - f(\bar{e}_2 e_1)\|^2 \\ D \|f(01) - f(e_2 \bar{e}_1)\|^2 & D \|f(11) - f(\bar{e}_2 \bar{e}_1)\|^2 \end{bmatrix} \quad (8.19)\end{aligned}$$

where $\bar{e} = 1 \oplus e$. The error state diagram for this TCM scheme is given in Fig. 8.18.

The matrix transfer function of the error state diagram is

$$\mathbf{G} = \mathbf{G}(10)[\mathbf{I}_2 - \mathbf{G}(11)]^{-1} \mathbf{G}(01) \quad (8.20)$$

where \mathbf{I}_2 is the 2×2 identity matrix. In this case there are only three error vectors possible, $\{01, 10, 11\}$.

From (8.19) we calculate

$$\mathbf{G}(01) = \frac{1}{2} \begin{bmatrix} D^2 & D^2 \\ D^2 & D^2 \end{bmatrix}, \quad \mathbf{G}(10) = \frac{1}{2} \begin{bmatrix} D^4 & D^4 \\ D^4 & D^4 \end{bmatrix}$$

$$\text{and } \mathbf{G}(11) = \frac{1}{2} \begin{bmatrix} D^2 & D^2 \\ D^2 & D^2 \end{bmatrix}$$

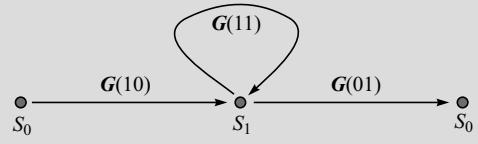


Fig. 8.18 The error state diagram.

Using (8.20) we obtain the matrix transfer function of the error state diagram as

$$\mathbf{G} = \frac{1}{2} \frac{D^6}{1-D^2} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad (8.21)$$

The scalar transfer function, $T(D)$, is then given by

$$T(D) = \frac{1}{2} \mathbf{1}^T \mathbf{G} \mathbf{1} = \frac{D^6}{1-D^2} \quad (8.22)$$

The upper bound on the probability of error can be computed by substituting $D = e^{-\left\{\frac{1}{4N_0}\right\}}$ in (8.22).

$$P_e \leq \frac{D^6}{1-D^2} \Big|_{D=e^{-\left\{\frac{1}{4N_0}\right\}}} \quad (8.23)$$

Example 8.12 Consider another rate $\frac{1}{2}$ TCM scheme with $m = 1$, and $M = 4$. The two state trellis diagram is same as the one in the previous example. However, the symbol allocation from the 4-PSK constellation is different, and is given Fig. 8.19.

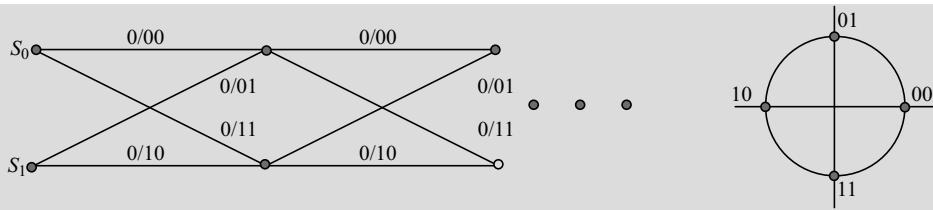


Fig. 8.19 Two-state trellis diagram.

Note that this symbol assignment violates the Ungerboeck design principles. Let us again denote the error vector by $\mathbf{e} = (e_2 \ e_1)$. The error state diagram for this TCM scheme is given in Fig. 8.20.

The matrix transfer function of the error state diagram is

$$\mathbf{G} = \mathbf{G}(11)[\mathbf{I}_2 - \mathbf{G}(10)]^{-1}\mathbf{G}(01) \quad (8.24)$$

where \mathbf{I}_2 is the 2×2 identity matrix. In this case there are only three error vectors possible, $\{01, 10, 11\}$. From (8.19) we calculate

$$\mathbf{G}(01) = \frac{1}{2} \begin{bmatrix} D^2 & D^2 \\ D^2 & D^2 \end{bmatrix}, \quad \mathbf{G}(10) = \frac{1}{2} \begin{bmatrix} D^4 & D^4 \\ D^4 & D^4 \end{bmatrix}, \text{ and } \mathbf{G}(11) = \frac{1}{2} \begin{bmatrix} D^2 & D^2 \\ D^2 & D^2 \end{bmatrix}$$

Using (8.24) we obtain the matrix transfer function of the error state diagram as

$$\mathbf{G} = \frac{1}{2} \frac{D^4}{1-D^4} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad (8.25)$$

The scalar transfer function, $T(D)$, is then given by

$$T(D) = \frac{1}{2} \mathbf{1}^T \mathbf{G} \mathbf{1} = \frac{D^4}{1-D^4} \quad (8.26)$$

The upper bound on the probability of error is

$$P_e \leq \left. \frac{D^4}{1-D^4} \right|_{D=e^{\frac{-1}{4N_0}}} \quad (8.27)$$

Comparing (8.23) and (8.27) we observe that simply by changing the symbol assignment to the branches of the trellis diagram we degrade the performance considerably. In the second example, the upper bound on the error probability has loosened by two orders of magnitude (assuming $D \ll 1$, i.e., for the high SNR case).

A tighter upper bound on the error event probability is given by (exercise)

$$P_e \leq \frac{1}{2} \operatorname{erfc} \left(\sqrt{\frac{d_{\text{free}}^2}{4N_0}} \right) e^{\frac{d_{\text{free}}^2}{4N_0}} T(D) \Big|_{D=e^{\frac{-1}{4N_0}}} \quad (8.28)$$

From (8.28) an asymptotic estimate on the error event probability can be obtained by considering only the error events with free Euclidean distance

$$P_e \approx \frac{1}{2} N(d_{\text{free}}) \operatorname{erfc} \left(\sqrt{\frac{d_{\text{free}}^2}{4N_0}} \right) \quad (8.29)$$

The bit error probability can be upper bounded simply by weighting the pairwise error probabilities by the number of incorrect input bits associated with each error vector and then dividing the result by m . Therefore,

$$P_b \leq \frac{1}{m} \left. \frac{\partial T(D, I)}{\partial I} \right|_{I=1, D=e^{\frac{-1}{4N_0}}}, \quad (8.30)$$

where $T(D, I)$ is the **Augmented Generating Function** of the modified state diagram. The concept of the modified state diagram was introduced in the chapter on Convolutional Codes. A tighter upper bound can also be obtained for the bit error probability, and is given by



$$P_e \leq \frac{1}{2m} \operatorname{erfc} \left(\sqrt{\frac{d_{\text{free}}^2}{4N_0}} \right) e^{\frac{d_{\text{free}}^2}{4N_0}} \left. \frac{\partial T(D, I)}{\partial I} \right|_{I=1, D=e^{\frac{-1}{4N_0}}}. \quad (8.31)$$

8.7 Computation of d_{free}

LO 2

We have seen that the *Euclidean* free distance, d_{free} , is the single most important parameter for determining how good a TCM scheme is for AWGN channels. It defines the asymptotic coding gain of the scheme. In Chapter 7 (Section 7.5) we saw that the generating function can be used to calculate the *Hamming* free distance d_{free} . The transfer function of the error state diagram, $T(D)$, includes information about the distance of all the paths in the trellis from the all zero path. If $T(D)$ is obtained in a closed form, the value of d_{free} follows immediately from the expansion of the function in a power series. The generating function can be written as

$$T(D) = N(d_{\text{free}}) D^{d_{\text{free}}^2} + N(d_{\text{next}}) D^{d_{\text{next}}^2} + \dots \quad (8.32)$$

where d_{next}^2 is the second smallest squared Euclidean distance. Hence, the smallest exponent of D in the series expansion is d_{free}^2 . However, in most cases, a closed form expression for $T(D)$ may not be available, and one has to resort to numerical techniques.

Next, consider the function

$$\phi_1(D) = \ln \left[\frac{T(eD)}{T(D)} \right] \quad (8.33)$$

$\phi_1(D)$ decreases monotonically to the limit d_{free}^2 as $D \rightarrow 0$. Therefore, we have an upper bound on d_{free}^2 provided $D > 0$. In order to obtain a lower bound on d_{free}^2 consider the following function

$$\phi_2(D) = \frac{\ln T(D)}{\ln D} \quad (8.34)$$

Taking logarithm on both sides of (8.32) we get,

$$d_{\text{free}}^2 \ln D = \ln T(D) - \ln N(d_{\text{free}}) - \ln \left[1 + \frac{N(d_{\text{free}})}{N(d_{\text{next}})} D^{d_{\text{next}}^2 - d_{\text{free}}^2} \dots \right] \quad (8.35)$$

If we take $D \rightarrow 0$, provided $D > 0$, from (8.34) and (8.35) we obtain

$$\frac{\ln T(D)}{\ln D} = d_{\text{free}}^2 - \varepsilon(D) \quad (8.36)$$

where, $\varepsilon(D)$ is a function that is greater than zero, and tends to zero monotonically as $D \rightarrow 0$. Thus, if we take smaller and smaller values of $\phi_1(D)$ and $\phi_2(D)$, we can obtain values that are extremely close to d_{free}^2 .

It should be kept in mind that even though d_{free}^2 is the single most important parameter to determine the quality of a TCM scheme, two other parameters are also influential.

- (i) The **error coefficient** $N(d_{\text{free}})$: A factor of two increase in this error coefficient reduces the coding gain by approximately 0.2 dB for error rates of 10^{-6} .
- (ii) The **next distance** d_{next} : is the second smallest Euclidean distance between two paths forming an error event. If d_{next} is very close to d_{free} , the SNR requirement for good approximation of the upper bound on P_e may be very large.

DID YOU KNOW ? So far, we have focussed primarily on AWGN channels. We found that the best design strategy is to maximise the free Euclidean Distance, d_{free} , for the code. In the next section, we consider the design rules for TCM over fading channels. Just to remind the readers, fading channels are frequently encountered in radio and mobile communications. One common cause of fading is the multipath nature of the propagation medium. In this case, the signal arrives at the receiver from different paths (with time varying nature) and gets added together. Depending on whether the signals from different paths add up in phase, or out of phase, the net received signal exhibits a random variation in amplitude and phase. The drops in the received signal amplitude (below a threshold) are called **fades**.

LO 2

In this section, we will consider the performance of trellis coded M -ary Phase Shift Keying (MPSK) over a fading channel. We know that a TCM encoder takes in an input bit stream and outputs a sequence of symbols. In this treatment we will assume that each of these symbols s_i belongs to the MPSK signal set. By using complex notation, each symbol can be represented by a point in the complex plane. The coded signals are **interleaved** in order to spread the burst of errors caused by the slowly varying fading process. These interleaved symbols are then pulse-shaped for no **inter-symbol interference** and finally translated to RF frequencies for transmission over the channel. The channel corrupts these transmitted symbols by adding a fading gain (which is a negative gain, or a positive loss, depending on one's outlook) and AWGN. At the receiver end the received sequences are demodulated and quantised for soft decision decoding. In many implementations, the channel estimator provides an estimate of the channel gain, which is also termed as the **channel state information**. Thus, we can represent the received signal at time i as

$$r_i = g_i s_i + n_i \quad (8.37)$$

where n_i is a sample of the zero mean Gaussian noise process with variance $N_0/2$ and g_i is the complex channel gain, which is also a sample of a complex Gaussian process with variance σ_g^2 . The complex channel gain can be explicitly written using the phasor notation as follows

$$g_i = a_i e^{j\phi_i} \quad (8.38)$$

where, a_i and ϕ_i are the amplitude and phase processes respectively. We now make the following assumptions:

- (i) The receiver performs coherent detection,
- (ii) The interleaving is ideal, which implies that the fading amplitudes are statistically independent and the channel can be treated as memoryless.

Thus, we can write

$$r_i = a_i s_i + n_i \quad (8.39)$$

We know that for a channel with a diffused multipath and no direct path the fading amplitude is **Rayleigh** distributed with pdf

$$p_A(a) = 2ae^{-a^2}, a \geq 0 \quad (8.40)$$

DID YOU
KNOW

In case there exists a direct path in addition to the multipath, **Rician fading** is observed. The pdf of the Rician fading amplitude is given by

$$p_A(a) = 2a(1+K)e^{-(K+a^2(K+1))} I_0(2a\sqrt{K(1+K)}) \quad (8.41)$$

where $I_0(\cdot)$ is the zero-order, modified Bessel function of the first kind and K is the Rician parameter defined as follows.

Definition 8.8 The **Rician Parameter**, K , is defined as the ratio of the energy of the direct component to the energy of the diffused multipath component. For the extreme case of $K = 0$, the pdf of the Rician distribution becomes the same as the pdf of the **Rayleigh distribution**.

We now look at the performance of the TCM scheme over a fading channel. Let $\mathbf{r}_l = (r_1, r_2, \dots, r_l)$ be the received signal. The maximum likely decoder, which is usually implemented by the Viterbi decoder, chooses the coded sequence that most likely corresponds to the received signals. This is achieved by computing a metric between the sequence of received signals, \mathbf{r}_l , and the possible transmitted signals, \mathbf{s}_l . As we have seen earlier, this metric is related to the conditional channel probabilities

$$m(\mathbf{r}_l, \mathbf{s}_l) = \ln p(\mathbf{r}_l | \mathbf{s}_l) \quad (8.42)$$

If the channel state information is being used, the metric becomes

$$m(\mathbf{r}_l, \mathbf{s}_l; \hat{\mathbf{a}}_l) = \ln p(\mathbf{r}_l | \mathbf{s}_l, \hat{\mathbf{a}}_l) \quad (8.43)$$

Under the assumption of ideal interleaving, the channel is memoryless and hence the metrics can be expressed as the following summations

$$m(\mathbf{r}_l, \mathbf{s}_l) = \sum_{i=1}^l \ln p(r_i | s_i) \quad (8.44)$$

and

$$m(\mathbf{r}_l, \mathbf{s}_l; \hat{\mathbf{a}}_l) = \sum_{i=1}^l \ln p(r_i | s_i, \hat{a}_i) \quad (8.45)$$

First, we consider the scenario where the channel state information is known, i.e., $\hat{a}_i = a_i$. The metric can be written as

$$m(r_i, s_i; \hat{a}_i) = -|r_i - a_i s_i|^2 \quad (8.46)$$

Therefore, the pairwise error probability is given by

$$P_2(s_l, \hat{s}_l) = E_{a_l}[P_2(s_l, \hat{s}_l | a_l)] \quad (8.47)$$

where

$$P_2(s_l, \hat{s}_l | a_l) = P[m(r_l, \hat{s}_l; a_l) \geq m(r_l, s_l; a_l) | a_l] \quad (8.48)$$

and E is the statistical expectation operator. Using the Chernoff bound, the pairwise error probability can be upper bounded as follows.

$$P_2(s_l, \hat{s}_l) \leq \prod_{i=1}^l \frac{1+K}{1+K + \frac{1}{4N_0} |s_i - \hat{s}_i|^2} \exp\left[-\frac{K \frac{1}{4N_0} |s_i - \hat{s}_i|^2}{1+K + \frac{1}{4N_0} |s_i - \hat{s}_i|^2} \right] \quad (8.49)$$

For high SNR, the above equation simplifies to

$$P_2(s_l, \hat{s}_l) \leq \prod_{i \in \eta} \frac{(1+K)e^{-K}}{\frac{1}{4N_0} |s_i - \hat{s}_i|^2} \quad (8.50)$$

where η is the set of all i for which $s_i \neq \hat{s}_i$. Let us denote the number of elements in η by l_η , then we can write

$$P_2(s_l, \hat{s}_l) \leq \frac{((1+K)e^{-K})^{l_\eta}}{\left(\frac{1}{4N_0}\right)^{l_\eta} d_p^2(l_\eta)} \quad (8.51)$$

where

$$d_p^2(l_\eta) \leq \prod_{i \in \eta} |s_i - \hat{s}_i|^2 \quad (8.52)$$

is the **Squared Product Distance** of the signals $s_i \neq \hat{s}_i$. The term l_η is called the **Effective Length** of the error event (s_l, \hat{s}_l) . A union bound on the error event probability P_e has already been discussed before. For the high SNR case, the upper bound on P_e can be expressed as

$$P_e \leq \sum_{l_\eta} \sum_{d_p^2(l_\eta)} \alpha(l_\eta, d_p^2(l_\eta)) \frac{((1+K)e^{-K})^{l_\eta}}{\left(\frac{1}{4N_0}\right)^{l_\eta} d_p^2(l_\eta)} \quad (8.53)$$

where $\alpha(l_\eta, d_p^2(l_\eta))$ is the average number of code sequences having the effective length l_η and the squared product distance $d_p^2(l_\eta)$. The error event probability is actually dominated by the smallest effective length l_η and the smallest product distance $d_p^2(l_\eta)$. Let us denote the smallest effective length l_η by L and the corresponding product distance by $d_p^2(L)$. The error event probability can then be asymptotically approximated by

$$P_e \approx \alpha(L, d_p^2(L)) \frac{((1+K)e^{-K})^L}{\left(\frac{1}{4N_0}\right)^L d_p^2(L)} \quad (8.54)$$



We make the following observations from (8.54).

- (i) The error event probability asymptotically varies with the L^{th} power of SNR. This is similar to what is achieved with a time diversity technique. Hence, L is also called the time diversity of the TCM scheme.
- (ii) The important TCM design parameters are the time diversity, L , and the product distance $d_p^2(L)$. This is in contrast to the free Euclidean distance parameter for AWGN channel.
- (iii) TCM codes designed for AWGN channels would normally fare poorly in fading channels and vice-versa.
- (iv) For large values of the Rician parameter, K , the effect of the free Euclidean distance on the performance of the TCM scheme becomes dominant.
- (v) At low SNR, again, the free Euclidean distance becomes important for the performance of the TCM scheme.

Thus, the basic *design rules* for TCMs for fading channels, at high SNR and for small values of K , are

- (i) maximise the effective length, L , of the code, and
- (ii) minimise the minimum product distance $d_p^2(L)$.

Consider a TCM scheme with effective length, L , and the minimum product distance $d_{p_1}^2(L)$. Suppose the code is redesigned to yield a minimum product distance, $d_{p_2}^2(L)$ with the same L . The increase in the coding gain due the increase in the minimum product distance is given by

$$\Delta g = \text{SNR}_1 - \text{SNR}_2 \Big|_{P_1 = P_2} = \frac{10}{L} \log \frac{d_{p_2}^2(L)}{d_{p_1}^2(L)} \frac{\alpha_1}{\alpha_2} \quad (8.55)$$

where α_i , $i = 1, 2$, is the average number of code sequences with effective length L for the TCM scheme i . We observe that for a fixed value of L , increasing the minimum product distance corresponding to a smaller value of L is more effective in improving the performance of the code.

So far we have assumed that the channel state information was available. A similar analysis as carried out for the case where channel state information was available can also be done when the information about the channel is unavailable. In the *absence* of channel state information, the metric can be expressed as

$$m(r_i, s_i; \hat{a}_i) = -|r_i - s_i|^2 \quad (8.56)$$

After some mathematical manipulations it can be shown that



$$P_2(s_i, \hat{s}_i) \leq \frac{\left(2e/l_\eta\right)^{l_\eta}}{\left(1/N_0\right)^{l_\eta}} \frac{\left(\sum_{i \in l_\eta} |s_i - \hat{s}_i|^2\right)^{l_\eta}}{d_p^4(l_\eta)} (1+K)^{l_\eta} e^{-l_\eta K} \quad (8.57)$$

Using arguments discussed earlier in this section, the error event probability P_e can be determined for this case when the channel state information is not available.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 2:

1. Consider the TCM scheme shown in Fig. 8.3. Suppose, the received vector is given by $r = s_7, s_5, s_1, s_3, s_1, s_6 \dots$. Find the transmitted sequence and the input bit-stream to the TCM encoder.
2. Consider the TCM scheme shown in Fig. 8.21. Suppose the all zero path was transmitted, and the decoder decodes the following sequence: $s_7, s_5, s_2, s_3, s_1, s_6$. Depict the error event in the trellis.

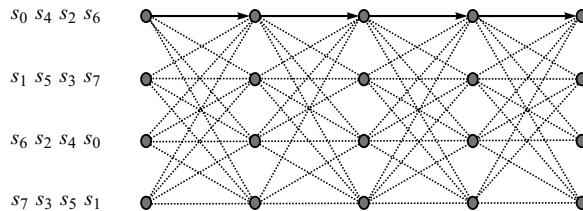


Fig. 8.21

If you have successfully solved the above problems,
you have mastered LO 2. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/618>



8.9 Space Time Trellis Codes

Space Time Trellis Codes (STTC) are coding techniques designed for multiple-antenna transmissions. STTCs are TCM schemes in which every branch of the trellis is labeled by N_t signals corresponding to the signals transmitted by the N_t transmit antennas.

Example 8.13 Figure 8.22 depicts a 2-STTC using 4-PSK. The edge label is denoted by c_1c_2 where c_1 is transmitted from antenna 1 and c_2 is transmitted from antenna 2.

LO 3

Underline the concept of Space Time Trellis Codes (STTC) and learn the STTC design criteria for fading channels.

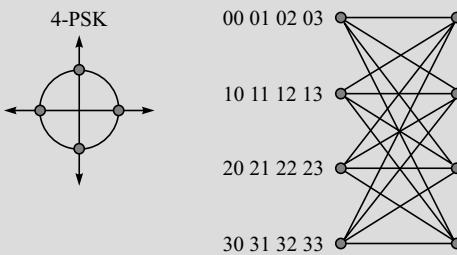


Fig. 8.22 A 2-STTC, 4-PSK, 4 states, 2 bits/s/Hz.

Example 8.14 Consider the MN -ary pulse position amplitude modulation (PPAM), which combines M -PAM and N -PPM. Let $M = 2$ and $N = 2$. The set partitioning, the symbol mapping and the encoder realization are shown in Fig. 8.23.

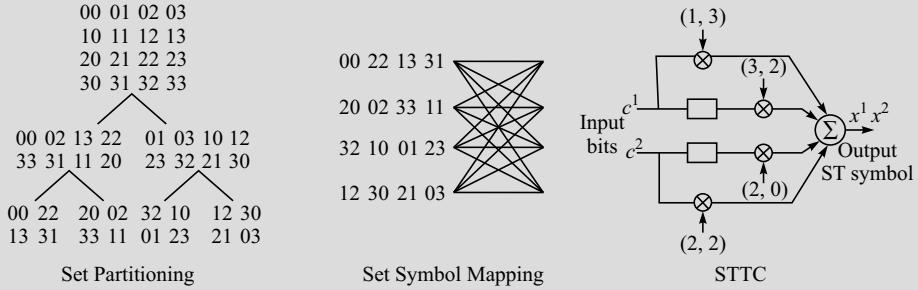


Fig. 8.23 Full rate, 2-state, 2,2PPAM STTC.

For decoding of a STTC code, **Vector Viterbi Algorithm** is required, assuming ideal channel state information at the receiver. Assuming that r_t^j is the received signal at receive antenna j at time t and α_{ij} is the path gain from transmit antenna i to receive antenna j , the branch metric for a transition labelled $q_t^1 q_t^2 \dots q_t^{N_t}$ is given by

$$\sum_{j=1}^{N_r} \left| r_t^j - \sum_{i=1}^{N_t} \alpha_{ij} q_t^i \right|^2 \quad (8.58)$$

The Viterbi algorithm is used to compute the path with the lowest accumulated metric.

Figure 8.24 shows a generic encoder structure for a full rate M -PSK STTC. This encoder structure would be valid in general for a full rate STTC with any other M -ary signal constellation. The encoder consists of $m = \log_2 M$ feedforward shift registers into which m binary input sequences c^1, c^2, \dots, c^m are fed. The multiplier coefficient set for the k^{th} shift register is denoted by

$$g^k = [(g_{0,1}^k, g_{0,2}^k, \dots, g_{0,N_t}^k), (g_{1,1}^k, g_{1,2}^k, \dots, g_{1,N_t}^k), \dots, (g_{v_1,1}^k, g_{v_1,2}^k, \dots, g_{v_1,N_t}^k)] \quad (8.59)$$

where $g_{j,i}^k$, $k = 1, 2, \dots, m$, $j = 1, 2, \dots, v_k$, $i = 1, 2, \dots, N_t$ is an element of M -PSK (or M -ary) constellation set and v_k is the memory order of the k^{th} shift register. The multiplier outputs from all shift registers are added modulo M , giving the encoder output $x = (x^1, x^2, \dots, x^{N_t})$. The total memory order of the encoder, denoted by v , is given by

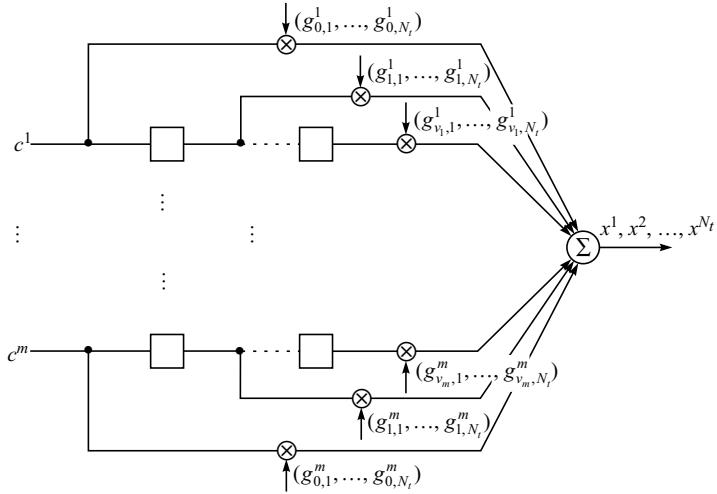


Fig. 8.24 Encoder for STTC.

$$v = \sum_{k=1}^m v_k \quad (8.60)$$

The value of v_k is determined by

$$v_k = \left\lfloor \frac{v+k-1}{\log_2 M} \right\rfloor \quad (8.61)$$

Let us now look at the performance criteria for STTC. We consider two scenarios: slow Rayleigh Fading and fast Rayleigh Fading.

Slow Rayleigh Fading

For each input symbol s_t , the space-time encoder generates N_t code symbols $c_t^1, c_t^2, \dots, c_t^{N_t}$, which are simultaneously transmitted from the N_t transmit antennas. We define the code vector as $\mathbf{c}_t = [c_t^1 c_t^2 \dots c_t^{N_t}]^T$. Suppose that the code vector sequence $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_J\}$ is transmitted. Consider the pairwise error probability (PEP) that the ML decoder decides erroneously in favour of the legitimate code vector sequence $\tilde{\mathcal{C}} = \{\tilde{\mathbf{c}}_1, \tilde{\mathbf{c}}_2, \dots, \tilde{\mathbf{c}}_J\}$. Consider a frame of length J and define the $N_t \times N_t$ error matrix \mathbf{A} as

$$\mathbf{A}(\mathcal{C}, \tilde{\mathcal{C}}) = \sum_{t=1}^J (\mathbf{c}_t - \tilde{\mathbf{c}}_t)(\mathbf{c}_t - \tilde{\mathbf{c}}_t)^* \quad (8.62)$$

If ideal channel state information is available at the receiver, then it is possible to show that the pairwise error probability (PEP) of erroneous detection is given by

$$P(\mathcal{C} \rightarrow \tilde{\mathcal{C}}) \leq \left(\prod_{i=1}^r \lambda_i \right)^{-N_r} (E_s / 4N_0)^{-rN_r} \quad (8.63)$$

where E_s is the symbol energy, N_0 is the noise power spectral density, N_r is the number of receive antennas, r is the rank of the error matrix \mathbf{A} and $\lambda_i, i=1, \dots, r$ are the non-zero eigenvalues of \mathbf{A} . Thus, a diversity

gain of rN_r and a coding gain of $\left(\prod_{i=1}^r \lambda_i\right)^{1/r}$ is obtained. Minimising maximum PEP leads to following design criteria.

Rank-determinant Criteria

1. *Rank criterion:* In order to achieve maximum diversity advantage, the matrix $A(C, \tilde{C})$ has to be full rank over all possible C and \tilde{C} .
2. *Determinant criterion:* Suppose $A(C, \tilde{C})$ is full rank for all C and \tilde{C} then, in order to achieve maximum coding advantage, the minimum determinant of $A(C, \tilde{C})$ over all possible C and \tilde{C} should be maximised.

Fast Rayleigh Fading

Let $\rho(C, \tilde{C})$ denote the set of time instances $1 \leq t \leq J$ such that $|c_t - \tilde{c}_t| \neq 0$ and n_H be the number of such time instances. It can be shown for fast Rayleigh fading that

$$P(C \rightarrow \tilde{C}) \leq \prod_{t \in \rho(C, \tilde{C})} (|c_t - \tilde{c}_t|^2 E_s / 4N_0)^{-N_r} \quad (8.64)$$

leading to the following design criteria for fast fading.

Product-distance Criteria

1. *Distance Criterion:* In order to achieve the diversity $n_H N_r$, we require for any two codewords C and \tilde{C} , that the minimum symbol-wise Hamming distance between C and \tilde{C} be at least n_H .
2. *Product Criterion:* To obtain maximum coding advantage, maximise the minimum of the products $\prod_{t \in \rho(C, \tilde{C})} |c_t - \tilde{c}_t|^2$ over all pairs of distinct codewords C and \tilde{C} .

Regarding STTC, we make the following observations:

- (i) The constraint length of an r -STTC is at least $r - 1$.
- (ii) For N_t transmit antennas and a diversity gain of r , the transmission rate R is upper bounded by $R \leq N_t - r + 1$. Thus, for full diversity gain ($r = N_t$), the maximum rate is full rate corresponding to a bandwidth efficiency of m bits/s/Hz for a 2^m signal constellation.
- (iii) If m is the transmission rate, the trellis complexity (number of states) is at least $2^{m(r-1)}$.

Example 8.15 Consider a wireless system with two transmit antennas ($N_t = 2$) to be deployed over a fast Rayleigh fading channel. It uses a 4-state STTC along with binary PAM. Let us use Ungerboeck's criteria for maximising the minimum Euclidean distance. The optimal binary-PAM STTC scheme based on product distance and Euclidean distance criteria is shown in Fig. 8.25.

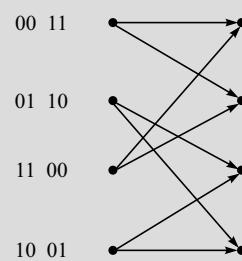


Fig. 8.25 Binary-PAM STTC based on product distance and Euclidean distance criteria.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 3:

1. Show that the encoder shown in Fig. 8.26 can be represented as a Space Time Trellis Code. If we use an 8-PSK, how many states will be there in the STTC?

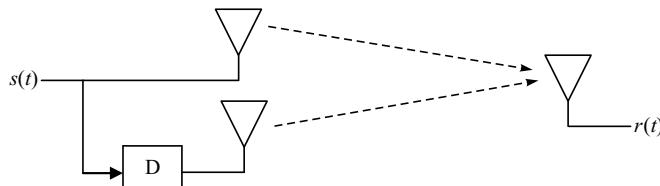


Fig. 8.26

2. Consider the Space Time Trellis Code shown in Fig. 8.27.

- (i) Which constellation is being employed?
- (ii) What is the rate of this STTC?
- (iii) What is the diversity provided by this code?

S
S
M

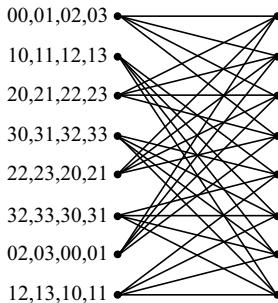


Fig. 8.27

If you have successfully solved the above problems,
you have mastered LO 3. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/619>



8.10 Concluding Remarks

Coding and modulation were first analysed together as a single entity by Massey in 1974. Prior to that time, in all coded digital communications systems, the encoder/decoder and the modulator/demodulator were

designed and optimised separately. Massey's idea of combined coding and modulation was concretised in the seminal paper by Ungerboeck in 1982. Similar ideas were also proposed earlier by Imai and Hirakawa in 1977, but did not get the due attention. The primary advantage of TCM was the ability to achieve increased power efficiency without the customary increase in the bandwidth introduced by the coding process. In the following years the theory of TCM was formalised by different researchers. Calderbank and Mazo showed that the asymmetric one-dimensional TCM schemes provide more coding gain than symmetric TCM schemes. Rotationally invariant TCM schemes were proposed by Wei in 1984, which were subsequently adopted by CCITT for use in the high speed voiceband modems. STTC are an extension of the conventional TCM schemes to multiantenna systems. These codes may be designed to extract diversity gain and coding gain.

LEARNING OUTCOMES

- The TCM Technique allows us to achieve a better performance *without* bandwidth expansion or using extra power.
- The minimum Euclidean Distance between any two paths in the trellis is called the free Euclidean Distance, d_{free} of the TCM scheme.
- The difference between the values of the SNR for the coded and uncoded schemes required to achieve the same error probability is known as the coding gain, $g = \text{SNR}|_{\text{uncoded}} - \text{SNR}|_{\text{coded}}$. At high SNR, the

coding gain can be expressed as $g_\infty = g|_{\text{SNR} \rightarrow \infty} = 10 \log \frac{(d_{\text{free}}^2 / E_s)_{\text{coded}}}{(d_{\text{free}}^2 / E_s)_{\text{uncoded}}}$, where g_∞ represents the Asymptotic Coding Gain and E_s is the average signal energy.

- The mapping by Set Partitioning is based on successive partitioning of the expanded 2^{m+1} -ary signal set into subsets with increasing minimum Euclidean Distance. Each time we partition the set, we reduce the number of the signal points in the subset, but increase the minimum distance between the signal points in the subset.
- Ungerboeck's TCM design rules (based on heuristics) for AWGN channels are:

Rule 1: Parallel transitions, if present, must be associated with the signals of the subsets in the lowest layer of the set partitioning tree. These signals have the minimum Euclidean Distance $\Delta_{\tilde{m}+1}$.

Rule 2: The transitions originating from or merging into one state must be associates with signals of the first step of set partitioning. The Euclidean distance between these signals is at least Δ_1 .

Rule 3: All signals are used with equal frequency in the Trellis Diagram.

- The Viterbi Algorithm can be used to decode the received symbols for a TCM scheme. The branch metric used in the decoding algorithm is the *Euclidean Distance* between the received signal and the signal associated with the corresponding branch in the trellis.
- The average number of nearest neighbours at free distance, $N(d_{\text{free}})$, gives the average number of paths in the trellis with free Euclidean Distance d_{free} from a transmitted sequence. This number is used in conjunction with d_{free} for the evaluation of the error event probability.
- The probability of error $P_e \leq T(D)|_{D=e^{-1/4N_0}}$, where, $T(D) = \frac{1}{N} \mathbf{1}^T \mathbf{G} \mathbf{1}$, and the matrix

$$\mathbf{G} = \sum_{l=1}^{\infty} \sum_{E_l \neq 0} \prod_{n=1}^l \mathbf{G}(e_n)$$

is the scalar transfer function. A tighter upper bound on the error event probability is given by

$$P_e \leq \frac{1}{2} \operatorname{erfc} \left(\sqrt{\frac{d_{\text{free}}^2}{4N_0}} \right) e^{\frac{d_{\text{free}}^2}{4N_0}} T(D) \Big|_{D=e^{\frac{-1}{4N_0}}}.$$

- For fading channels, $P_e(s_l, \hat{s}_l) \leq \frac{(1+K)e^{-K}}{\left(\frac{1}{4N_0}\right)^{l_\eta} d_p^2(l_\eta)}$ where $d_p^2(l_\eta) \leq \prod_{i \in \eta} |s_i - \hat{s}_i|^2$. The term l_η is

the effective length of the error event (s_l, \hat{s}_l) and K is the Rician parameter. Thus, the error event probability is dominated by the smallest effective length l_η and the smallest product distance $d_p^2(l_\eta)$.

- The design rules for TCMs for fading channels, at high SNR and for small values of K , are
 - (i) maximising the effective length, L , of the code, and
 - (ii) minimising the minimum product distance $d_p^2(L)$.
- The increase in the coding gain due to increase in minimum product distance is given by

$$\Delta g = \text{SNR}_1 - \text{SNR}_2 \Big|_{P_{e_1} - P_{e_2}} = \frac{10}{L} \log \frac{d_{p_2}^2(L)}{d_{p_1}^2(L)} \frac{\alpha_1}{\alpha_2},$$
 where α_i , $i = 1, 2$, is the average number of code sequences with effective length L for the TCM scheme i .
- Space Time Trellis Codes (STTC) are coding techniques designed for multiple-antenna transmissions. STTCs are TCM schemes in which every branch of the trellis is labeled by N_t signals corresponding to the signals transmitted by the N_t transmit antennas.
- For decoding of a STTC code, vector Viterbi algorithm is used, assuming ideal channel state information at the receiver.
- The design rule for slow Rayleigh Fading channel is given by the Rank-determinant criteria.
 - (i) *Rank criterion*: In order to achieve maximum diversity advantage, the matrix $A(C, \tilde{C})$ has to be full rank over all possible C and \tilde{C} .
 - (ii) *Determinant criterion*: Suppose $A(C, \tilde{C})$ is full rank for all C and \tilde{C} then, in order to achieve maximum coding advantage, the minimum determinant of $A(C, \tilde{C})$ over all possible C and \tilde{C} should be maximised.
- The design rule for fast Rayleigh Fading channel is given by the Product-distance criteria.
 - (i) *Distance criterion*: In order to achieve the diversity $n_H N_r$, we require for any two codewords C and \tilde{C} , that the minimum symbol-wise Hamming distance between C and \tilde{C} be at least n_H .
 - (ii) *Product criterion*: To obtain maximum coding advantage, maximise the minimum of the products

$$\prod_{t \in \rho(C, \tilde{C})} |c_t - \tilde{c}_t|^2$$
 over all pairs of distinct codewords C and \tilde{C} .

A little inaccuracy sometimes saves a ton of explanation.

H.H. Munro (Saki) (1870–1916)



MULTIPLE CHOICE QUESTIONS

- 8.1 The TCM technique allows us to achieve a better performance without

 - (a) Using extra power
 - (b) Using extra bandwidth
 - (c) Both (a) and (b)
 - (d) None of the above.

8.2 The d_{free} of a TCM scheme is

 - (a) The minimum Euclidean Distance from the all zero path in the trellis
 - (b) The minimum Euclidean Distance between any two paths in the trellis
 - (c) The minimum Hamming Distance between any two paths in the trellis
 - (d) None of the above.

8.3 The coding gain can be calculated using $g|_{SNR \rightarrow \infty} =$

 - (a) $10 \log \frac{(d_{free}^2 / E_s)_{\text{uncoded}}}{(d_{free}^2 / E_s)_{\text{coded}}}$
 - (b) $10 \log \frac{(d_{free} / E_s)_{\text{coded}}}{(d_{free} / E_s)_{\text{uncoded}}}$
 - (c) $20 \log \frac{(d_{free}^2 / E_s)_{\text{coded}}}{(d_{free}^2 / E_s)_{\text{uncoded}}}$
 - (d) $10 \log \frac{(d_{free}^2 / E_s)_{\text{coded}}}{(d_{free}^2 / E_s)_{\text{uncoded}}}.$

8.4 In Ungerboeck's TCM design rules:

 - (a) Parallel transitions, if present, must be associated with the signals of the subsets in the lowest layer of the set partitioning tree
 - (b) The transitions originating from or merging into one state must be associates with signals of the first step of set partitioning
 - (c) All signals are used with equal frequency
 - (d) All of the above.

8.5 TCM decoding is typically carried out using

 - (a) Ungerboeck Algorithm
 - (b) Vandermonde Algorithm
 - (c) Viterbi Algorithm
 - (d) None of the above.

8.6 For Viterbi decoding, the distance between symbols of the received sequence and the reference sequence is measured in terms of

 - (a) Manhattan distance
 - (b) Euclidean distance
 - (c) Hamming distance
 - (d) Jensen-Shannon distance.

8.7 For designing trellis code for AWGN channels, the emphasis must be on maximising

 - (a) Euclidean distance between code vectors
 - (b) Hamming distance between code vectors
 - (c) Both (a) and (b)
 - (d) None of the above.

8.8 For designing trellis code for fading channels, the emphasis must be on maximising

 - (a) Euclidean distance between code vectors
 - (b) Effective length, L
 - (c) Both (a) and (b)
 - (d) None of the above.

8.9 For designing trellis code for fading channels, the emphasis must be on minimising

 - (a) The minimum product distance
 - (b) The maximum product distance
 - (c) The free distance
 - (d) None of the above.

8.10 The design rule for Space Time Trellis Codes for slow Rayleigh fading channel is given by

- | | |
|------------------------|-------------------------------|
| (a) The Rank criterion | (b) The Determinant criterion |
| (c) Both (a) and (b) | (d) None of the above |

For interactive quiz with answers, scan the QR code given here

Or

Visit <http://qrcode.flipick.com/index.php/573>



SHORT-ANSWER TYPE QUESTIONS



- 8.1 TCM techniques allow us to achieve a better performance *without* bandwidth expansion or using extra power. True or false?
- 8.2 TCM schemes are linear. True or false?
- 8.3 To determine the d_{free}^2 for a TCM scheme, we find the Euclidean distance of all possible paths from the all-zero path and choose the minimum. True or false?
- 8.4 What is the Euclidean distance between two closest points for 8-PSK in terms of the E_s ?
- 8.5 A 64-state TCM scheme has a $d_{free}^2 = 6.34E_s$. What is the asymptotic coding gain with respect to the uncoded minimum squared Euclidean distance of $2E_s$?
- 8.6 Suppose we incorporate a TCM scheme with $d_{free}^2 = 3E_s$ in a wireless handset that requires charging every 18 hours. If the uncoded d_{free}^2 was $2E_s$, what would be the new time between charging?
- 8.7 What is the best possible asymptotic coding gain for a 4-state trellis with 8 PSK modulation?
- 8.8 The best possible g_∞ for an 8-state TCM scheme with 8-PSK is 4 dB. True or false?
- 8.9 As per Ungerboeck's design rules, all signals are used with equal frequency in the Trellis Diagram. True or false?
- 8.10 What metric is used by the Viterbi decoder for decoding TCM?
- 8.11 $N(d_{free})$, the average number of paths in the trellis with free Euclidean Distance d_{free} from a transmitted sequences, is used in conjunction with d_{free} for the evaluation of the error event probability. True or false?
- 8.12 Name a telecommunication standard that uses TCM schemes.
- 8.13 Which algorithm is used for decoding of a STTC code?
- 8.14 Give the STTC design rules for slow Rayleigh Fading channel.
- 8.15 Give the STTC design rules for fast Rayleigh Fading channel.

For answers, scan the QR code given here

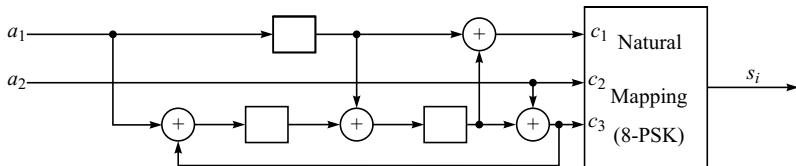
Or

Visit <http://qrcode.flipick.com/index.php/620>




PROBLEMS

- S** 8.1 Suppose natural mapping is performed in a TCM scheme, i.e., $s_0 \leftrightarrow 000, s_1 \leftrightarrow 001, \dots, s_7 \leftrightarrow 111$. Find the d_{free}^2 between the all zero path and the path generated by the sequence 000 011 010 101 100 000.
- M** 8.2 Consider a rate 2/3 convolutional code defined by
- $$\mathbf{G}(D) = \begin{bmatrix} 1 & D & D+D^2 \\ D^2 & 1+D & 1+D+D^2 \end{bmatrix} \quad (8.65)$$
- This code is used with an 8PSK signal set that uses Gray coding (the three bits per symbol are assigned such that the codes for two adjacent symbols differ only in 1 bit location). The throughput of this TCM scheme is 2 bits/sec/Hz.
- (i) How many states are there in the trellis diagram for this encoder?
 - (ii) Find the free Euclidean distance.
 - (iii) Find the asymptotic coding gain with respect to uncoded QPSK, which has a throughput of 2 bits/sec/Hz.
- S** 8.3 In Problem 8.2, suppose instead of Gray coding, natural mapping is performed, i.e., $s_0 \leftrightarrow 000, s_1 \leftrightarrow 001, \dots, s_7 \leftrightarrow 111$.
- (i) Find the free Euclidean distance.
 - (ii) Find the asymptotic coding gain with respect to uncoded QPSK (2 bits/sec/Hz).
- D** 8.4 Consider the TCM encoder shown in Fig. 8.28.

**Fig. 8.28**

- (i) Draw the state diagram for this encoder.
 - (ii) Draw the trellis diagram for this encoder.
 - (iii) Find the free Euclidean distance, d_{free}^2 . In the trellis diagram, show one pair of two paths which result in d_{free}^2 . What is $N(d_{\text{free}}^2)$?
 - (iv) Next, use set partitioning to assign the symbols of 8-PSK to the branches of the trellis diagram. What is the d_{free}^2 now?
 - (v) Encode the following bit stream using this encoder: 1 0 0 1 0 0 0 1 0 1 0 ... Give your answer for both the natural mapping and mapping using set partitioning.
 - (vi) Compare the asymptotic coding gains for the two different kinds of mapping.
- M** 8.5 We want to design a TCM scheme that has a 2/3 convolutional encoder followed by a signal mapper. The mapping is done based on set partitioning of the irregular constellation diagram shown in Fig. 8.29. The trellis is a four-state, fully connected trellis.
- (i) Perform set partitioning for the following asymmetric constellation diagram.

- (ii) What is the free Euclidean distance, d_{free}^2 , for this asymmetric TCM scheme? Compare it with the d_{free}^2 for the case when we use the standard 8-PSK signal constellation.
- (iii) How will you choose the value of θ for improving the performance of the TCM scheme using the asymmetric signal constellation?

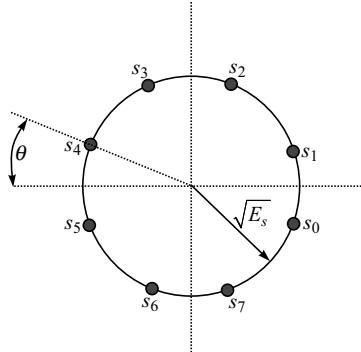


Fig. 8.29

- D 8.6 Consider the rate $\frac{3}{4}$ encoder shown in Fig. 8.30. The four output bits from the encoder are mapped onto one of the sixteen possible symbols from the constellation diagram shown below. Use Ungerboeck's design rules to design a TCM scheme for an AWGN channel. What is the asymptotic coding gain with respect to uncoded 8-PSK?

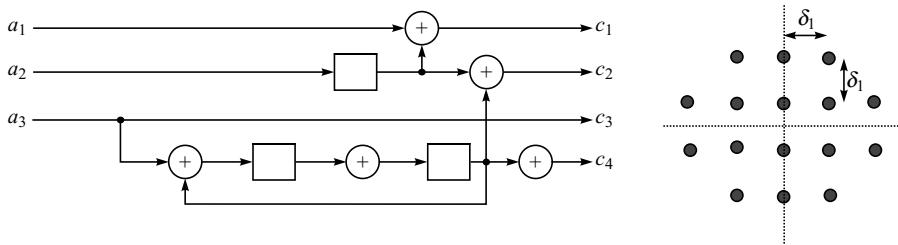


Fig. 8.30

- M 8.7 Consider the expression for pairwise error probability over a Rician fading channel

$$P_2(s_l, \hat{s}_l) \leq \prod_{i=1}^l \frac{1+K}{1+K + \frac{1}{4N_0} |s_i - \hat{s}_i|^2} \exp \left[-\frac{K \frac{1}{4N_0} |s_i - \hat{s}_i|^2}{1+K + \frac{1}{4N_0} |s_i - \hat{s}_i|^2} \right] \quad (8.66)$$

- (i) Show that the above inequality can be simplified for a large value of K to

$$P_2(s_l, \hat{s}_l) \leq \prod_{i=1}^l \exp \left[-\frac{1}{4N_0} |s_i - \hat{s}_i|^2 \right] \quad (8.67)$$

Comment.

- (ii) Show that for low SNR the original inequality may be expressed as

$$P_2(s_l, \hat{s}_l) \leq \exp\left[-\frac{d_E^2(s_l, \hat{s}_l)}{4N_0}\right]. \quad (8.68)$$

- M** 8.8 Consider a TCM scheme designed for a Rician fading channel with an effective length L and the minimum product distance $d_p^2(L)$. Suppose, we wish to redesign this code to obtain an improvement of 3 dB in SNR.
- (i) Compute the desired effective length L if the $d_p^2(L)$ is kept unchanged.
 - (ii) Compute the desired product distance $d_p^2(L)$ if the effective length L is kept unchanged.
- S** 8.9 Suppose you have to design a TCM scheme for an AWGN channel ($\text{SNR} = \gamma$). The desired BER is P_e . Draw a flowchart as to how you will go about designing such a scheme.
- (i) How many states will there be in your Trellis?
 - (ii) How will you design the convolutional encoder?
 - (iii) Would you have parallel paths in your design?
 - (iv) What kind of modulation scheme will you choose and why?
 - (v) How will you assign the symbols of the modulation scheme to the branches?
- S** 8.10 For Viterbi decoding the metric used is of the form
- $$m(r_l, s_l) = \ln p(r_l | s_l) \quad (8.69)$$
- (i) What is the logic behind choosing such a metric?
 - (ii) Suggest another metric that will be suitable for fading channels. Give reasons for your answer.
- S** 8.11 A TCM scheme designed for a Rician Fading Channel ($K = 3$) and a high SNR environment ($\text{SNR} = 20 \text{ dB}$) has $L = 5$ and $d_p^2(L) = 2.34E_s^2$. It has to be redesigned to produce an improvement of 2 dB. What is the $d_p^2(L)$ of the new code? What can you comment on the new d_{free}^2 .
- D** 8.12 Consider the TCM scheme shown in Fig. 8.31 consisting of a rate $\frac{1}{2}$ convolutional encoder coupled with a mapper.
- (i) Draw the trellis diagram for this encoder.
 - (ii) Determine the scalar transfer function, $T(D)$.
 - (iii) Determine the augmented generating function, $T(D, L, I)$.
 - (iv) What is the minimum Hamming distance (d_{free}^H) of this code?
 - (v) How many paths are there with this d_{free}^H ?

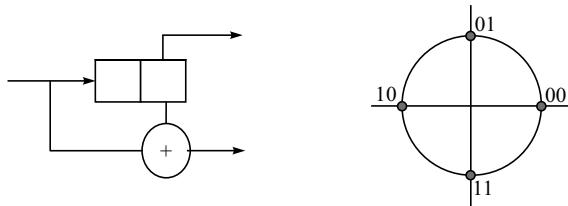


Fig. 8.31

- M** 8.13 Consider the pairwise error probability $P_2(s_l, \hat{s}_l)$.

- (i) For a maximum likelihood decoder, prove that

$$P_2(s_l, \hat{s}_l) = \int_r f(\mathbf{r}) p_{R|S}(\mathbf{r} | s_l) d\mathbf{r} \quad (8.70)$$

where \mathbf{r} is the received vector, $p_{R|S}(\mathbf{r} | s_l)$ is the channel transition probability density function and

$$f(\mathbf{r}) = \begin{cases} 1 & \text{if } p_{R|S}(\mathbf{r} | \hat{s}_l) \geq p_{R|S}(\mathbf{r} | s_l) \\ 0 & \text{otherwise} \end{cases} \quad (8.71)$$

- (ii) Show that

$$f(\mathbf{r}) \leq \frac{p_{R|S}(\mathbf{r} | \hat{s}_l)}{p_{R|S}(\mathbf{r} | s_l)}, \quad (8.72)$$

and

$$P_2(s_l, \hat{s}_l) \leq \int_r \sqrt{p_{R|S}(\mathbf{r} | \hat{s}_l) p_{R|S}(\mathbf{r} | s_l)} d\mathbf{r} \quad (8.73)$$

- M** 8.14 Consider the two TCM schemes depicted in Fig. 8.32. Both the schemes use a 4-state fully connected trellis together with 8 PSK.

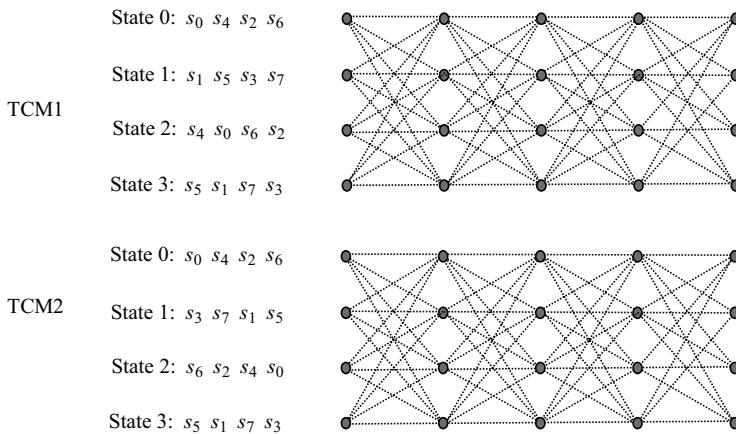


Fig. 8.32

- (i) Which of the two schemes adheres to the Ungerboeck's TCM design rules?
- (ii) Find the d_{free}^2 and $d_p^2(L)$ for these schemes.
- (iii) Which one is more suitable for a purely AWGN channel? Which one is more suitable for a fading channel? Comment on your answer.

Note: TCM1 is called the **Wilson Leung Code**.

- M** 8.15 Consider the constellation diagram shown in Fig. 8.33 prescribed in the ITU V.17 standard. Carry out set partitioning on this constellation diagram.

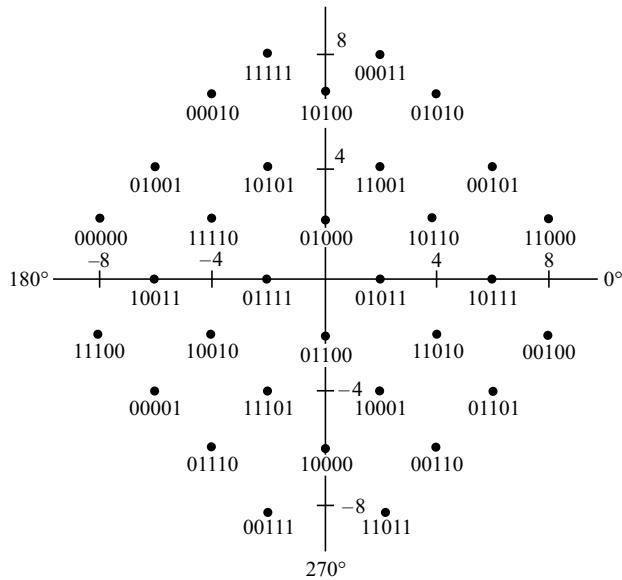


Fig. 8.33

COMPUTER PROBLEMS



- 8.1 Write a computer program to perform trellis coded modulation, given the trellis structure and the mapping rule. The program should take in an input bit stream and output a sequence of symbols. The input to the program may be taken as two matrices, one that gives the connectivity between the states of the trellis (essentially the structure of the trellis) and the second, which gives the branch labels.
- 8.2 Write a computer program to calculate the squared free Euclidean distance d_{free}^2 , the effective length L , and the minimum product distance, $d_p^2(L)$, of a TCM scheme, given the trellis diagram and the labels on the branches.
- 8.3 Write a computer program that performs Viterbi decoding on an input stream of symbols. This program makes use of a given trellis and the labels on the branches of the trellis diagram.
- 8.4 Verify the performance of the different TCM schemes given in this chapter in AWGN environment. To do so, take a long chain of random bits and input it to the TCM encoder. The encoder will produce a sequence of symbols (analogue waveforms). Corrupt these symbols with AWGN of different noise power, i.e., simulate scenarios with different SNRs. Use Viterbi decoding to decode the received sequence of corrupted symbols (distorted waveforms). Generate a plot of the BER versus the SNR and compare it with the theoretically predicted error rates.
- 8.5 Write a computer program to observe the effect of decoding window size for the Viterbi decoder. Generate a plot of the error rate versus the window size. Also plot the number of computations versus the window size.

- 8.6 Write a computer program that performs exhaustive search in order to determine a rate 2/3 TCM encoder which is designed for AWGN (maximise d_{free}^2). Assume that there are four states in the trellis diagram and it is a fully connected trellis. The branches of this trellis are labelled using the symbols from an 8-PSK signal set. Modify the program to perform exhaustive search for a good TCM scheme with a four-state trellis with the possibility of parallel branches.
- 8.7 Draw the family of curves depicting the relation between P_e and l_η for different values of K (Rician Parameter) for
- (i) High SNR,
 - (ii) Low SNR.
- Comment on the plots.
- 8.8 Write a computer program to generate a table for TCM codes with maximum d_{free}^2 for 16PSK. The table should list TCM schemes with different number of states (4 – 512). The program should carry out optimal symbol mapping in order to maximise d_{free}^2 . Plot the g_∞ versus the number of states. Comment on your result.
- 8.9 Write a computer program to generate a table for TCM codes with maximum d_{free}^2 for MQAM. The table should list TCM schemes with different number of states (8 – 256) and $M = 16, 32, 64, 128$. The program should carry out optimal symbol mapping in order to maximise d_{free}^2 .

PROJECT IDEAS

- 8.1 **TCM for Rayleigh Fading Channels:** Write a computer program that performs exhaustive search in order to determine a rate 2/3 TCM encoder which is designed for a Rayleigh fading channel. Assume that there are four states in the trellis diagram and it is a fully connected trellis. The branches of this trellis are labelled using the symbols from an 8-PSK signal set. List out the $d_p^2(L)$ and L of some of the better codes found during the search.
- 8.2 **Space Time Trellis Code Design for AWGN:** Design an STTC with 16 states and 32 states using 8-PSK that provides a diversity order of 2. Test these codes over AWGN channel with 2 transmit and 1 receive antennas. Plot the BER versus SNR of the coded system, as well as, the uncoded system. Compare the slopes of the curves and comment on your observation.
- 8.3 **Space Time Trellis Code Design for Fading Channels:** Write a computer program to exhaustively search for optimal 4-state, fully connected STTC for
- (i) Slow Rayleigh fading channels.
 - (ii) Fast Rayleigh fading channels.
- Carry out the search for MPSK and MQAM signal constellations.
- 8.4 **Trellis Coded Spatial Modulation (TCSM):** We wish to combine the concept of TCM with multi-antenna spatial modulation. The multiple transmit antennas can be considered as additional constellation points for the trellis and the entire set of transmit antennas can be sub-divided into subsets such that the spacing between antennas within a particular sub-set is maximised. Design a 4-state, 8PSK-based, TCSM scheme for a 4×4 MIMO system and compute the asymptotic coding gain. How does the TCSM perform in a Rician fading channel with a Rician parameter, K ? How will you combine TCSM with the Alamouti code?

- 8.5 Super-Orthogonal Space Time Trellis Codes (SO-STTC):** In SO-STTC we combine space-time block codes with a trellis code to come up with a new structure that guarantees full diversity. Design SO-STTC for two transmit antennas and one receive antenna. Carry out simulations for fading channels and compare your results with those of the existing space-time trellis codes.
- 8.6 STTC-MIMO-OFDM:** Consider the Space Time Trellis Code-Multi Input Multi Output-Orthogonal Frequency Division Multiplexing (STTC-MIMO-OFDM) communication system shown in Fig. 8.34. Design appropriate STTCs and carry out simulations for fading channels for various choices of STTCs. Determine the diversity gain and the coding gain provided by the STTC-MIMO-OFDM communication system.



Fig. 8.34

REFERENCES FOR FURTHER READING

- Divsalar, E., D. McLane, P.J. and Simon, M.K.; *Introduction to Trellis-Coded Modulation with Applications*, Macmillan, 1991.
- Lin, S. and Costello, D.J.; *Error Control Coding: Fundamentals and Applications* (2nd edition), Prentice-Hall, 2004.

PART - IV

Coding for Secure Communications

Chapter 9 Cryptography

Chapter 10 Physical Layer Security

Cryptography

One cannot escape the feeling that these mathematical formulae have an independent existence and an intelligence of their own, that they are wiser than we are, wiser even than their discoverers, that we get more out of them than we originally put in to them.

Heinrich Hertz (1857–1894)

Learning objectives

After studying this chapter the students will be able to:

- LO 1** Know the basic operations used by common encryption techniques.
- LO 2** Explain the concept of secret-key cryptography, Data Encryption Standard (DES), International Data Encryption Algorithm (IDEA) and RC Ciphers.
- LO 3** Underline the concept of public-key cryptography and implement the Rivest, Shamir and Adleman (RSA) algorithm, Pretty Good Privacy (PGP) and one-way Hashing.
- LO 4** Understand and solve problems related to the Diffie-Hellman protocol, elliptic curve cryptography, quantum cryptography, biometric encryption and chaos-based cryptography.

9.1 Introduction to Cryptography

Cryptography is the science of devising methods that allow information to be sent in a secure form in such a way that the only person able to retrieve this information is the intended recipient. Encryption is based on algorithms that scramble information into unreadable or non-discernable form. Decryption is the process of restoring the scrambled information to its original form (see Fig. 9.1).

DID YOU KNOW ? A **Cryptosystem** is a collection of algorithms and associated procedures for hiding and revealing (un-hiding!) information. **Cryptanalysis** is the process (actually, the art) of analysing a crypto-

...
This chapter comes with a video overview by the author. Scan here to know more or
Visit <http://qrcode.flipick.com/index.php/627>



system, either to verify its integrity or to break it for ulterior motives. An **Attacker** is a person or system that performs cryptanalysis in order to break a cryptosystem. Attackers are also referred to as hackers, interlopers or eavesdroppers. The process of attacking a cryptosystem is often called **Cracking** or **Hacking**.

The job of a **Cryptanalyst** is to find the weaknesses in the cryptosystem. In many cases, the developers of a cryptosystem announce a public challenge with a large prize-money for anyone who can crack the scheme. Once a cryptosystem is broken (and the cryptanalyst discloses his techniques), the designers of the scheme try to strengthen the algorithm. Just because a cryptosystem has been broken does not render it useless. The hackers may have broken the system under optimal conditions using equipment (fast computers, dedicated microprocessors, etc.) that is usually not available to regular people. Some cryptosystems are rated in terms of the number of years and the price of the computing equipment it would take to break them!



In the last few decades cryptographic algorithms, being mathematical by nature, have become sufficiently advanced that they can only be handled by computers. This, in effect, means that the uncoded message (prior to encryption) is binary in form and can, therefore, be anything—a picture, a voice, an e-mail or even a video.

DID YOU KNOW ?

Cryptography is not merely used for military and diplomatic communications as many people tend to believe. In reality, cryptography (although obviously essential for the military and diplomatic services) has many commercial uses and applications. From protecting confidential company information, to protecting a telephone call, to allowing someone to order a product on the Internet without the fear of their credit card number being intercepted and used against them, cryptography is all about increasing the level of privacy of individuals and groups. For example, cryptography is often used to prevent forgers from counterfeiting winning lottery tickets. Each lottery ticket can have two numbers printed onto it, one plaintext and one the corresponding cipher. Unless the counterfeiter has cryptanalysed the lottery's cryptosystem he or she will not be able to print an acceptable forgery.

In this chapter

We begin with an overview of different encryption techniques. We will then introduce the basic operations used by common encryption techniques, in LO 1.

We will then understand the concept of secret-key cryptography. Some specific secret-key cryptographic techniques will be studied. Specifically, the Data Encryption Standard (DES) and the International Data Encryption Algorithm (IDEA) will be discussed in detail. A brief introduction to RC ciphers will also be given, in LO 2.

The public-key cryptography will be introduced next and the Rivest, Shamir and Adleman (RSA) algorithm will be discussed as a popular example. A hybrid cryptographic algorithm and Pretty Good Privacy (PGP) will be introduced. We will also learn about one-way Hashing, in LO 3.

Finally, a flavour of some other cryptographic techniques in use today will be given. The chapter will discuss the Diffie-Hellman Protocol and the Elliptic Curve Cryptography. The domain of cryptography is ever-expanding, and this chapter will introduce some of the emerging techniques such as Quantum Cryptography, Biometric Encryption and Chaos-based cryptography. The chapter will conclude with a discussion on Cryptanalysis and the politics of cryptography, in LO 4.

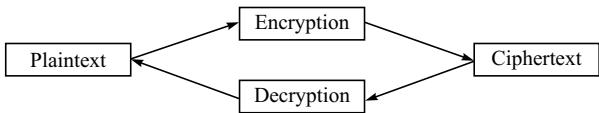


Fig. 9.1 The process of encryption and decryption.

9.2 An Overview of Encryption Techniques

The goal of a cryptographic system is to provide a high level of confidentiality, integrity, non-repudiation and authenticity to information that is exchanged over networks.

Confidentiality provides privacy for messages and stored data by hiding information using encryption techniques. Message integrity provides assurance to all parties that a message remains unchanged from the time it was created to the time it was opened by the recipient. Altered messages can lead to actions that are inappropriate. Non-repudiation can provide a way of proving that the message came from someone even if they try to deny it. Authentication provides two services. Firstly, it identifies the origin of the message. Secondly, it verifies the identity of a user logging into a system and continues to verify their identity in case someone tries to break into the system.

LO 1



Know the basic operations used by common encryption techniques.



Definition 9.1 A message being sent is known as **Plaintext**. The message is coded using a cryptographic algorithm. This process is called **Encryption**. An encrypted message is known as **Ciphertext**, and is turned back into plaintext by the process of **Decryption**.

It must be assumed that any eavesdropper has access to all communications between the sender and the recipient. A method of encryption is only secure even if with such complete access, the eavesdropper is still unable to recover the original plaintext from the ciphertext.

There is a big difference between *security* and *obscurity*. Suppose, a message is left for somebody in an airport locker and the details of the airport is also known. If the locker number is known only to the intended recipient, then this message is not secure, merely obscure. If however, all potential eavesdroppers know the exact location of the locker, and they still cannot open the locker and access the message, then this message is secure.

Definition 9.2 A **key** is a value that causes a cryptographic algorithm to run in a specific manner and produce a specific ciphertext as an output. The key size is usually measured in bits. The bigger the key size, the more secure will be the algorithm.

Example 9.1 Suppose we have to encrypt and send the following stream of binary data (which might be originating from voice, video, text or any other source)

0 1 1 0 0 0 1 0 1 0 0 1 1 1 1 1 ...

We can use a 4-bit long key, $x = 1011$, to encrypt this bit stream. To perform encryption, the plaintext (binary bit stream) is first subdivided into blocks of 4 bits.

0 1 1 0 0 0 1 0 1 0 0 1 1 1 1 ...

Each sub-block is XORed (binary addition without carry) with the key, $x = 1011$. The basic XOR operation (\oplus) is as follows.

$$0 \oplus 0 = 0, 0 \oplus 1 = 1, 1 \oplus 0 = 1, 1 \oplus 1 = 0.$$

The encrypted message will be

1 1 0 1 1 0 0 1 0 0 1 0 0 1 0 0 ...

The recipient must also possess the knowledge of the key in order to decrypt the message. The decryption process is fairly simple in this case. The ciphertext (the received binary bit stream) is first

subdivided in to blocks of 4 bits. Each sub-block is XORed with the key, $x = 1011$. The decrypted message will be the original plaintext

0 1 1 0 0 0 1 0 1 0 0 1 1 1 1 1...

It should be noted that just one key is used both for encryption and decryption.

Example 9.2 Let us devise an algorithm for text messages, which we shall call *character + x*. Let $x = 5$. In this encryption technique, we replace every alphabet by the fifth one following it, i.e., A becomes F, B becomes G, C becomes H, and so on. The recipients of the encrypted message just need to know the value of the key, x , in order to decipher the message. The key must be kept separate from the encrypted message being sent. Because there is just one key which is used for encryption and decryption, this kind of technique is called **Symmetric Cryptography** or **Single Key Cryptography** or **Secret Key Cryptography**. The problem with this technique is that the key has to be kept confidential. Also, the key must be changed from time to time to ensure secrecy of transmission. This means that the secret key (or the set of keys) has to be communicated to the recipient. This might be done physically.

To get around this problem of communicating the key, the concept of **Public Key Cryptography** was developed by Diffie and Hellman. This technique is also called the **Asymmetric Encryption**. The concept is simple. There are two keys, one is held privately and the other one is made public. What one key can lock, the other key can unlock.

Example 9.3 Suppose we want to send an encrypted message to recipient A using the public key encryption technique. To do so we will use the public key of the recipient A and use it to encrypt the message. After recipient A receives the message, he decrypts it with his private key. Only the private key of recipient A can decrypt a message that has been encrypted with his public key. Similarly, recipient B can only decrypt a message that has been encrypted with his public key. Thus, no private key ever needs to be communicated and hence one does not have to trust any communication channel to convey the keys.

INDUSTRIAL RELEVANCE



Let us consider another scenario, where we want to send somebody a message and also provide a proof that the message is actually from us (a lot of harm can be done by providing bogus information, or rather, misinformation!). In order to keep a message private and also provide the authentication (that it is indeed from us), we can perform a special encryption on the plain text with our *private* key, then encrypt it again with the *public* key of the recipient. The recipient uses his private key to open the message and then use our public key to verify the authenticity. This technique is said to use **Digital Signatures**. The FIPS 186 standard is the **Digital Signature Standard** (DSS), which specifies the **Digital Signature Algorithm** (DSA).

There is another important encryption technique called the **One-way Function**. It is basically a non-reversible quick encryption method. The encryption is easy and fast, but the decryption is not. Suppose

we send a document to recipient A and want to check at a later time whether the document has been tampered with. We can do so by running a one-way function, which produces a fixed length value called a **Hash** (also called the **Message Digest**). The hash is the unique signature of the document that can be sent along with the document. Recipient A can run the same one-way function to check whether the document has been altered.

The actual mathematical function used to encrypt and decrypt messages is called a cryptographic algorithm or cipher. This is only part of the system used to send and receive secure messages. This will become clearer further on when specific systems are discussed in detail.

As with most historical ciphers, the security of the message being sent relied on the algorithm itself remaining secret. This technique is known as a **Restricted Algorithm**. These have the following fundamental drawbacks:

- (i) The algorithm obviously has to be restricted to only those people that you want to be able to decode your message. Therefore, a new algorithm must be invented for every discrete group of users.
- (ii) A large or changing group of users cannot utilise them, as every time one user leaves the group, everyone must change algorithms.
- (iii) If the algorithm is compromised in any way, a new algorithm must be implemented.

Because of these drawbacks, restricted algorithms are no longer popular, and have given way to key based algorithms.

Practically all modern cryptographic systems make use of a key. Algorithms that use a key system allow all details of the respective algorithms to be widely available. This is because all of the security lies in the key. With a key-based algorithm the plaintext is encrypted and decrypted by the algorithm which uses a certain key, and the resulting ciphertext is dependent on the key, and not the algorithm. This means that an eavesdropper can have a complete copy of the algorithm in use, but without the specific key used to encrypt that message it is useless. A **nonce** is an arbitrary number that may be used only once in cryptographic algorithms, e.g., a timestamp, a counter, or a random number. A nonce differs with each transaction. It is often a random or pseudo-random number used in an authentication protocol to ensure that old communications cannot be reused for carrying out replay attacks.

This chapter deals with cryptographic **Algorithms**. The word *algorithm* comes from the name of the ninth century Persian mathematician, Abu Abdullah Muhammad ibn Musa al-Khwarizmi (780–850 A.D.), whose works introduced Indian numerals and algebraic concepts. Two general principles which guide the design of practical algorithms are *diffusion* and *confusion*. Diffusion means spreading out of the influence of a single plaintext digit over many ciphertext digits so as to hide the statistical structure of the plaintext. An extension of this idea is to spread the influence of a single key digit over many digits of ciphertext. Confusion means use of transformations which complicate dependence of the statistics of ciphertext on the statistics of plaintext.



According to **Kerckhoff's Principle**, a cryptosystem should remain secure even if everything about the system, except the key, is known. We will assume for our discussions that the adversary will know how the encryption and decryption algorithms work.

LO 1

9.3 Operations used by Encryption Algorithms

Although the methods of encryption/decryption have changed dramatically since the advent of computers, there are still only two basic operations that can be carried out on a piece of plaintext—substitution and transposition. The only real difference is that, earlier these were carried out with the alphabet; nowadays they are carried out on binary bits.

Substitution

Substitution operations replace bits in the plaintext with other bits decided upon by the algorithm, to produce ciphertext. This substitution then just has to be reversed to produce plaintext from ciphertext. This can be made increasingly complicated. For instance one plaintext character could correspond to one of a number of ciphertext characters (homophonic substitution), or each character of plaintext is substituted by a character of corresponding position in a length of another text (running cipher).

Example 9.4 Julius Caesar was one of the first to use substitution encryption to send messages to troops during the war. The substitution method he invented advances each character three spaces in the alphabet. Thus,

$$\text{THIS IS SUBSTITUTION CIPHER} \quad (9.1)$$

becomes

$$\text{WKLV LU VXEVWLWXWLRQ FLSKHU} \quad (9.2)$$

This is an example of **Caesar cipher**.

Example 9.5 The **Vigenère cipher** is a method of substitution encryption by using a series of different Caesar ciphers based on the letters of a keyword. A Vigenère cipher uses a Vigenère square as shown below.

Plaintext→ ↓Key	A	B	C	D	...	W	X	Y	Z
A	A	B	C	D	...	W	X	Y	Z
B	B	C	D	E	...	X	Y	Z	A
C	C	D	E	F	...	Y	Z	A	B
:	:	:	:	:	...	:	:	:	:
X	X	Y	Z	A	...	T	U	V	W
Y	Y	Z	A	B	...	U	V	W	X
Z	Z	A	B	C	...	V	W	X	Y

Let us choose the key ‘BAY’. Our key has only 3 letters, while we need to encrypt seven letters of the plaintext. So, we repeat our key as follows so that there are seven letters: B A Y B A Y B. Each one of the letters in the key will be used to encrypt (and decrypt) each corresponding letter in the plaintext. In order to encrypt, we choose the plaintext letter in the top row of the Vigenère square and choose the key letter in the first column. The ciphertext is the entry in the Vigenère square corresponding to the column of the plaintext and the row of the key. Using the Vigenère square above, we carry out the following encryption.

Plaintext Character	Key Character	Ciphertext Character
A	B	B
B	A	B
A	Y	Y
D	B	E
C	A	C
A	Y	Y
B	B	C

Thus, the ciphertext becomes: B B Y E C Y C.

Transposition

Transposition (or permutation) does not alter any of the bits in plaintext, but instead move their positions around within it. If the resultant ciphertext is then put through more transpositions, the end result is increasingly secure.

XOR

XOR is an exclusive-or operation. It is a Boolean operator such that if one of two bits is true, then so is the result, but if both are true or both are false then the result is false.

For example,

$$\begin{aligned} 0 \text{ XOR } 0 &= 0 \\ 1 \text{ XOR } 0 &= 1 \\ 0 \text{ XOR } 1 &= 1 \\ 1 \text{ XOR } 1 &= 0 \end{aligned} \tag{9.3}$$

A surprising amount of commercial software uses simple XOR functions to provide security, including the USA digital cellular telephone network and many office applications, and it is trivial to crack. However the XOR operation, as will be seen later in this paper, is a vital part of many advanced cryptographic algorithms when performed between long blocks of bits that also undergo substitution and/or transposition.

Example 9.6 Let us consider a **Multiplicative cipher**, which is a slightly more sophisticated technique than the Caesar cipher. Instead of adding a key, we multiply by the key number. Suppose, the key is, $k = 3$. We follow these steps for enciphering.

1. Replace each letter by the number it represents.
2. Multiply each number by the key, $k = 3$. Use modulo 26 arithmetic.
3. Replace the resulting number by letter equivalent.

Thus, for $k = 3$, we will have:

Plain	A	B	C	D	E	...	Y	Z
Number	1	2	3	4	5	...	25	26
$\times k$	3	6	9	12	15	...	75	78
Mod(26)	3	6	9	12	15	...	23	0/26
Cipher	C	F	I	L	O	...	W	Z

Note that, in order for this cipher to work, the key, k , and the modulus, n , should be relatively prime.

 **Example 9.7** Let us consider an **Affine cipher**. Recall that an affine transform is of the type $Y = aX + b$. For an Affine cipher, we follow these steps for enciphering.

1. Replace each letter by the number it represents.
2. Multiply each number by a and then add b . Use modulo 26 arithmetic.
3. Replace the resulting number by a letter equivalent.

Note that we can, alternately, add b' first and then multiply by a' .

<i>Plain</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	...	<i>Y</i>	<i>Z</i>
Number	1	2	3	4	5	...	25	26
+ 3	4	5	6	7	8		2	3
× 5	20	25	30	35	40	...	10	15
Mod(26)	20	25	4	9	14	...	10	15
Cipher	T	Y	D	I	N	...	J	O



Definition 9.3 The **Avalanche effect** is a property of any encryption algorithm such that a small change in either the plaintext or the key produces a significant change in the ciphertext.

In the absence of the avalanche effect, a cryptanalyst can possibly make predictions about the input, given only the output. This may enable him/her to partially or completely break the cryptographic algorithm. Thus, the avalanche effect is a desirable condition for any cryptographic algorithm.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 1:

- How many one-to-one affine ciphers can be constructed for the English alphabet? M
- What is the total number of unique keys possible for the Playfair cipher? Do account for the trivial cases also. M
- Consider a multiplicative cipher with $k = 2$. Build the cipher table and explain the problem with this cipher. M
- In a message encrypted by the Affine cipher, the trigram NHM appears maximum number of times. Can you deduce the keys for this Affine cipher? D

If you have successfully solved the above problems,
you have mastered LO 1. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/622>



Levels of Difficulty

S **Simple:** Level 1 and Level 2 Category

M **Medium:** Level 3 and Level 4 Category

D **Difficult:** Level 5 and Level 6 Category

9.4 Symmetric (Secret Key) Cryptography

Symmetric Algorithms (or **Single Key Algorithms** or **Secret Key Algorithms**) have one key that is used both to encrypt and decrypt the message, hence their name. In order for the recipient to decrypt the message they need to have an identical copy of the key. This presents one major problem, the distribution of the keys. Unless the recipient can meet the sender in person and obtain a key, the key itself must be transmitted to the recipient, and is thus susceptible to eavesdropping. However, single key algorithms are fast and efficient, especially if large volumes of data need to be processed.

In symmetric cryptography, the two parties that exchange messages use the same algorithm. Only the key is changed from time to time. The same plaintext with a different key results in a different ciphertext. The encryption algorithm is available to the public, hence should be strong and well tested. The more powerful the algorithm, the lesser likely that an attacker will be able to decrypt the resulting cipher.

The size of the key is critical in producing a strong ciphertext. The U.S. National Security Agency (NSA) stated in the mid-1990s that a 40-bit length was acceptable to them (i.e., they could crack it sufficiently quickly!). Increasing processor speeds, combined with loosely-coupled multi-processor configurations, have brought the ability to crack such short keys within the reach of possible hackers. In 1998, it was suggested that in order to be strong, the key size needs to be at least 56 bits long. It was argued by an expert group as early as 1996 that 90 bits is a more appropriate length. Today, the most secure schemes use at least 128-bit keys, but commonly 256-bit keys, or even longer are preferred.

Symmetric cryptography provides a means of satisfying the requirement of message content security, because the content cannot be read without the secret key. There remains a risk of exposure, however, because neither party can be sure that the other party has not exposed the secret key to a third party (whether accidentally or intentionally).



Definition 9.4 *The **Playfair cipher** is a symmetric key encryption technique and is a digraph substitution cipher. Instead of single letters, as in the simple substitution cipher, the Playfair cipher encrypts pairs of letters (digraphs).*

This Playfair cipher technique is relatively harder to break than the simple substitution cipher. This is because the simple frequency analysis used for the substitution ciphers does not work with it. The frequency analysis of digraphs is possible, but considerably more difficult (well ‘more difficult’ is a relative term!). For the English alphabet, there are 600 possible digraphs rather than the 26 possible monographs. The Playfair algorithm is based on the use of a 5×5 matrix of letters constructed using a keyword.

Symmetric cryptography can also be used to address the integrity and authentication requirements. The sender creates a summary of the message, or **Message Authentication Code (MAC)**, encrypts it with the secret key, and sends that with the message. The recipient then re-creates the MAC, decrypts the MAC that was sent, and compares the two. If they are identical, then the message that was received must have been identical with that which was sent.

DID YOU KNOW ? As mentioned earlier, a major difficulty with symmetric schemes is that the secret key has to be possessed by both parties, and hence has to be transmitted from whoever creates it to the other party. Moreover, if the key is compromised, all of the message transmission security measures are undermined. The steps taken to provide a secure mechanism for creating and passing on the secret key are referred to as **Key Management**.

LO 2



Explain the concept of secret-key cryptography, Data Encryption Standard (DES), International Data Encryption Algorithm (IDEA) and RC Ciphers.

The technique does not adequately address the non-repudiation requirement, because both parties have the same secret key. Hence each is exposed to the risk of fraudulent falsification of a message by the other, and a claim by either party not to have sent a message is credible, because the other may have compromised the key.

There are two types of symmetric algorithms, block ciphers and stream ciphers.

Definition 9.5 Block Ciphers usually operate on groups of bits called blocks. Each block is processed a multiple number of times. In each round the key is applied in a unique manner. The more the number of iterations, the longer is the encryption process, but results in a more secure ciphertext.

Example 9.8 A **Feistel cipher** is a symmetric cryptographic structure used in the construction of block ciphers. It is named after the German-born cryptographer Horst Feistel. The Feistel structure has the advantage that both the encryption and decryption operations are very similar, thereby substantially reducing the size of the code or hardware required to implement such a cipher. In the classic Feistel structure, one half of the data block is used to modify the other half of the data block. Subsequently, the halves are swapped.

Definition 9.6 Stream Ciphers operate on plaintext one bit (or one byte) at a time. Plaintext is streamed as raw bits through the encryption algorithm. While a block cipher will produce the same ciphertext from the same plaintext using the same key, a stream cipher will not. The ciphertext produced by a stream cipher will vary under the same conditions.

Example 9.9 Let us look at a simple stream cipher. Consider a plaintext stream $P = P_1 P_2 P_3 \dots$, and a key stream $K = k_1 k_2 k_3 \dots$

The encryption is done as follows: $C_1 = E(P_1, k_1)$, $C_2 = E(P_2, k_2)$, $C_3 = E(P_3, k_3)$, ...

The resulting ciphertext stream from the stream cipher would be $C = C_1 C_2 C_3 \dots$

INDUSTRIAL RELEVANCE



Example 9.10 Consider the stream cipher A5/1 used in the Global System for Mobile (GSM) communication. The A5/1 stream cipher uses three Linear Feedback Shift Registers (LFSRs) with irregular clocking. The details of the three LFSRs are given below.

LFSR number	Length in bits	Feedback polynomial
1	19	$x^{19} + x^{18} + x^{17} + x^{14} + 1$
2	22	$x^{22} + x^{21} + 1$
3	23	$x^{23} + x^{22} + x^{21} + x^8 + 1$

The registers are clocked in a stop-and-go fashion using a majority rule. Each register has an associated clocking bit. At each cycle, the clocking bit of all three registers is examined and the majority bit is determined. If the clocking bit agrees with the majority bit, the register is clocked.

The A5/1 is known to be a weak cipher and several attacks have been published.

Let us now consider a basic question: How long should a key be? There is no single answer to this question. It depends on the specific situation. To determine how much security one needs, the following questions must be answered:

1. How much is the data to be protected worth?
2. How long does it need to be secure?
3. What are the resources available to the cryptanalyst/hacker?

A customer list might be worth Rs. 1000, an advertisement data might be worth Rs. 50,000 and the master key for a digital cash system might be worth millions. In the world of stock markets, the secrets have to be kept for a couple of minutes. In the newspaper business today's secret is tomorrow's headlines. The census data of a country have to be kept secret for months (if not years). Corporate trade secrets are interesting to rival companies and military secrets are interesting to rival militaries. Thus, the security requirements can be specified in these terms. For example, one may require that the key length must be such that there is a probability of 0.0001% that a hacker with the resources of Rs. 1 million could break the system in 1 year, assuming that the technology advances at a rate of 25% per annum over that period. The minimum key requirements for different applications are listed in Table 9.1. This table should be used as a guideline only.

Table 9.1 Minimum key requirements for different applications

Type of information	Lifetime	Minimum key length
Tactical military information	Minutes/hours	56-64 bits
Product announcements	Days/weeks	64 bits
Interest rates	Days/weeks	64 bits
Trade secrets	decades	112 bits
Nuclear bomb secrets	> 50 years	128 bits
Identities of spies	> 50 years	128 bits
Personal affairs	> 60 years	> 128 bits
Diplomatic embarrassments	> 70 years	> 256 bits

Future computing power is difficult to estimate. A rule of thumb is that the efficiency of computing equipment divided by price doubles every 18 months, and increases by a factor of 10 every five years. Thus, in 50 years the fastest computer will be 10 billion times faster than today's! These numbers refer to the general-purpose computers. We cannot predict what kind of specialised crypto-system breaking computers might be developed in the years to come.

Two symmetric algorithms, both block ciphers, will be discussed in this chapter. These are the **Data Encryption Standard (DES)** and the **International Data Encryption Algorithm (IDEA)**.

9.5 Data Encryption Standard (DES)

LO 2



INDUSTRIAL RELEVANCE

DES, an acronym for the Data Encryption Standard, is the name of the Federal Information Processing Standard (FIPS) 46-3, which describes the data encryption algorithm (DEA). The DEA is also defined in the ANSI standard X9.32. ANSI X9.9 standard specifies a DES-based message authentication code (MAC) algorithm for wholesale banking. ANSI X9.23 standard addresses message formatting and representation issues related to the use of DES encryption in wholesale banking transactions.

Created by IBM, DES came about due to a public request by the US National Bureau of Standards (NSB) requesting proposals for a standard cryptographic algorithm that satisfied the following criteria.

- (i) Provides a high level of security
- (ii) The security depends on keys, not the secrecy of the algorithm
- (iii) The security is capable of being evaluated
- (iv) The algorithm is completely specified and easy to understand
- (v) It is efficient to use and adaptable
- (vi) Must be available to all users
- (vii) Must be exportable

DEA is essentially an improvement of the algorithm Lucifer developed by IBM in the early 1970s. The US National Bureau of Standards published the DES in 1975. While the algorithm was basically designed by IBM, the NSA and NBS (now NIST) played a substantial role in the final stages of the development. The DES has been extensively studied since its publication and is the best known and widely used symmetric algorithm in the world.

The DEA has a 64-bit block size and uses a 56-bit key during execution (8 parity bits are stripped off from the full 64-bit key). The DEA is a symmetric cryptosystem, specifically a 16-round Feistel cipher and was originally designed for implementation in hardware. When used for communication, both sender and receiver must know the same secret key, which can be used to encrypt and decrypt the message, or to generate and verify a message authentication code (MAC). The DEA can also be used for single-user encryption, such as to store files on a hard disk in encrypted form. In a multi-user environment, secure key distribution may be difficult; public-key cryptography provides an ideal solution to this problem.

NIST has re-certified DES (FIPS 46-1, 46-2, 46-3) every five years. FIPS 46-3 reaffirms DES usage as of October 1999, but single DES is permitted only for legacy systems. FIPS 46-3 includes a definition of triple-DES (TDEA, corresponding to X9.52). DES and triple-DES have now been replaced with the Advanced Encryption Standard (AES).

DES has now been in world-wide use for over 30 years, and due to the fact that it is a defined standard means that any system implementing DES can communicate with any other system using it, DES is used in banks and businesses all over the world, as well as in networks (as Kerberos) and to protect the password file on UNIX Operating Systems (as CRYPT).

DES Encryption

DES is a symmetric, block-cipher algorithm with a key length of 64 bits, and a block size of 64 bits (i.e., the algorithm operates on successive 64 bit blocks of plaintext). Being symmetric, the same key is used for encryption and decryption, and DES also uses the same algorithm for encryption and decryption.

First a transposition is carried out according to a set table (the initial permutation), the 64-bit plaintext block is then split into two 32-bit blocks, and 16 identical operations called rounds are carried out on each half. The two halves are then re-joined together, and the reverse of the initial permutation is carried out. The purpose of the first transposition is not clear, as it does not affect the security of the algorithm, but is thought to be for the purpose of allowing plaintext and ciphertext to be loaded into 8-bit chips in byte-sized pieces.

In any round, only one half of the original 64-bit block is operated on. The rounds alternate between the two halves. One round in DES consists of the following.

Key transformation

The 64-bit key is reduced to 56 by removing every eighth bit (these are sometimes used for error checking). Sixteen different 48-bit sub-keys are then created—one for each round. This is achieved by splitting the 56-bit key into two halves, and then circularly shifting them left by 1 or 2 bits, depending on the round. After this, 48 of the bits are selected. Because they are shifted, different groups of key bits are used in each subkey. This process is called a compression permutation due to the transposition of the bits and the reduction of the overall size.

Expansion permutation

After the key transformation, whichever half of the block is being operated on undergoes an expansion permutation. In this operation, the expansion and transposition are achieved simultaneously by allowing the first and fourth bits in each 4 bit block to appear twice in the output, i.e., the fourth input bit becomes the fifth and seventh output bits (see Fig 9.2).

The expansion permutation achieves 3 things: Firstly, it increases the size of the half-block from 32 bits to 48, the same number of bits as in the compressed key subset, which is important as the next operation is to XOR the two together. Secondly, it produces a longer string of data for the substitution operation that subsequently compresses it. Lastly, and most importantly, because in the subsequent substitutions the first and fourth bits appear in two S-boxes (described shortly), they affect two substitutions. The effect of this is that the dependency of the output bits on the input bits increases rapidly, and so does the security of the algorithm.

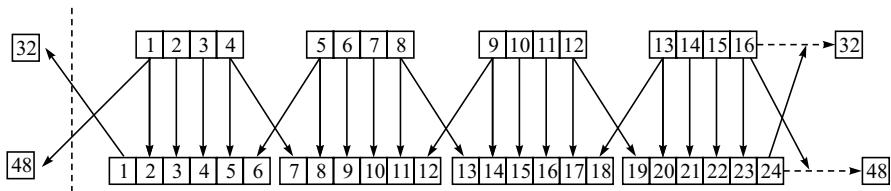


Fig. 9.2 The expansion permutation.

XOR

The resulting 48-bit block is then XORed with the appropriate subset key for that round.

Substitution

The next operation is to perform substitutions on the expanded block. There are eight substitution boxes, called S-boxes (see Fig 9.3). The first S-box operates on the first 6 bits of the 48-bit expanded block, the 2nd

S-box on the next six, and so on. Each S-box operates from a table of 4 rows and 16 columns, each entry in the table is a 4-bit number. The 6-bit number the S-box takes as input is used to look up the appropriate entry in the table in the following way. The 1st and 6th bits are combined to form a 2-bit number corresponding to a row number, and the 2nd to 5th bits are combined to form a 4-bit number corresponding to a particular column. The net result of the substitution phase is eight 4-bit blocks that are then combined into a 32-bit block.

It is the non-linear relationship of the S-boxes that really provide DES with its security, all the other processes within the DES algorithm are linear, and as such relatively easy to analyse.

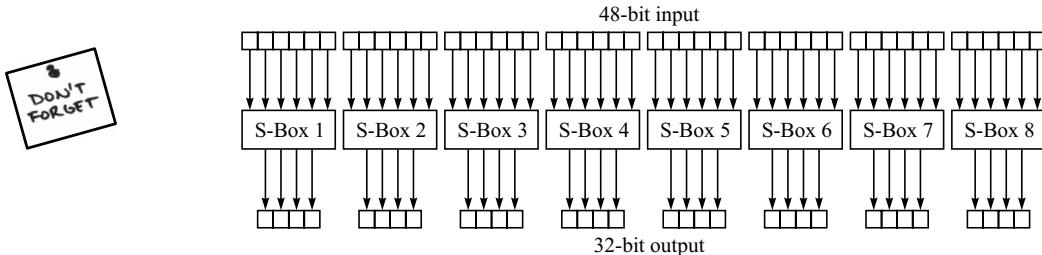


Fig. 9.3 The S-box substitution.

Permutation

The 32-bit output of the substitution phase then undergoes a straightforward transposition using a table sometimes known as the P-box.

After all the rounds have been completed, the two ‘half-blocks’ of 32 bits are recombined to form a 64-bit output, the final permutation is performed on it, and the resulting 64-bit block is the DES encrypted ciphertext of the input plaintext block.

DES Decryption

Decrypting DES is very easy (if one has the correct key!). Thanks to its design, the decryption algorithm is identical to the encryption algorithm—the only alteration that is made, is that to decrypt DES ciphertext, the subsets of the key used in each round are used in reverse, i.e., the sixteenth subset is used first.

Security of DES

Unfortunately, with advances in the field of cryptanalysis and the huge increase in available computing power, DES is no longer considered to be very secure. There are algorithms that can be used to reduce the number of keys that need to be checked, but even using a straightforward brute-force attack and just trying every single possible key there are computers that can crack DES in a matter of minutes. It is rumoured that the US National Security Agency (NSA) can crack a DES encrypted message in 3–15 minutes.

If a time limit of 2 hours to crack a DES encrypted file is set, then you have to check all possible keys (2^{56}) in 2 hours, which is roughly 5 trillion keys per second. While this may seem like a huge number, consider that a \$10 Application-Specific Integrated Circuits (ASICs) chip can test 200 million keys per second, and many of these can be paralleled together. It is suggested that a \$10 million investment in ASICs would allow a computer to be built that would be capable of breaking a DES encrypted message in 6 minutes.

DES can no longer be considered a sufficiently secure algorithm. If a DES-encrypted message can be broken in minutes by supercomputers today, then the rapidly increasing power of computers means that it will be a trivial matter to break DES encryption in the future (when a message encrypted today may still need to be secure). An extension of DES called DESX is considered to be virtually immune to an exhaustive key search.

Advanced Encryption Standard (AES)

AES is a symmetric-key block cipher. It is based on the **Rijndael cipher** developed by two cryptographers, Joan Daemen and Vincent Rijmen. Rijndael is a family of ciphers with different key and block sizes. AES is a symmetric-key algorithm, where the same key is used for both encrypting and decrypting the data. For AES, three types of Rijndael ciphers are used, each with a block size of 128 bits, but three different key lengths: 128, 192 and 256 bits. Rijndael cipher was designed to have the following characteristics.

- (i) Resistance against known attacks
- (ii) Speed and efficient coding over a wide range of platforms
- (iii) Design simplicity

AES is based on the principle known as a substitution-permutation network, where a combination of substitution and permutation is used. AES does not use the Feistel cipher structure. It may be recalled that in the classic Feistel structure, one half of the data block is used to modify the other half of the data block. Subsequently, the halves are swapped. In AES, the entire data block is processed in parallel during each round using substitutions and permutations.

9.6 International Data Encryption Algorithm (IDEA)

LO 2

IDEA was created in its first form by Xuejia Lai and James Massey in 1990, and was called the Proposed Encryption Standard (PES). In 1991, Lai and Massey strengthened the algorithm against differential cryptanalysis and called the result Improved PES (IPES). The name of IPES was changed to International Data Encryption Algorithm (IDEA) in 1992. IDEA is perhaps best known for its implementation of PGP (Pretty Good Privacy).

The Algorithm

IDEA is a symmetric, block-cipher algorithm with a key length of 128 bits, a block size of 64 bits, and as with DES, the same algorithm provides encryption and decryption.

IDEA consists of eight rounds using 52 sub-keys. Each round uses six sub-keys, with the remaining four being used for the output transformation. The sub-keys are created as follows.

Firstly, the 128-bit key is divided into eight 16-bit keys to provide the first eight sub-keys. The bits of the original key are then shifted 25 bits to the left, and then it is again split into eight sub-keys. This shifting and then splitting is repeated until all 52 sub-keys (SK1-SK52) have been created.

The 64-bit plaintext block is firstly split into four blocks (B1-B4). A round then consists of the following steps (OB stands for output block).

- OB1 = B1 * SK1 (multiply 1st sub-block with 1st sub-key)
- OB2 = B2 + SK2 (add 2nd sub-block to 2nd sub-key)
- OB3 = B3 + SK3
- OB4 = B4 * SK4 (multiply 3rd sub-block with 3rd sub-key)
- OB5 = OB1 XOR OB3 (XOR results of steps 1 and 3)
- OB6 = OB2 XOR OB4
- OB7 = OB5 * SK5 (multiply result of step 5 with 5th sub-key)
- OB8 = OB6 + OB7 (add results of steps 5 and 7)
- OB9 = OB8 * SK6 (multiply result of step 8 with 6th sub-key)
- OB10 = OB7 + OB9

OB11 = OB1 XOR OB9 (XOR results of steps 1 and 9)

OB12 = OB3 XOR OB9

OB13 = OB2 XOR OB10

OB14 = OB4 XOR OB10

The input to the next round is the four sub-blocks OB11, OB13, OB12 and OB14.

After the eighth round, the four final output blocks (F1-F4) are used in a final transformation to produce four sub-blocks of ciphertext (C1-C4) that are then re-joined to form the final 64-bit block of ciphertext.

C1 = F1 * SK49

C2 = F2 + SK50

C3 = F3 + SK51

C4 = F4 * SK52

Ciphertext = C1 & C2 & C3 & C4.

Security provided by IDEA

Not only is IDEA approximately twice as fast as DES, but it is also considerably more secure. Using a brute-force approach, there are 2^{128} possible keys. If a billion chips that could each test 1 billion keys a second were used to try and crack an IDEA-encrypted message, it would take them 10^{13} years which is considerably longer than the age of the universe! Being a fairly new algorithm, it is possible that a better attack than brute-force, which, when coupled with much more powerful machines in the future may be able to crack a message. However, for a long way into the future, IDEA seems to be an extremely secure cipher.

9.7 RC Ciphers

LO 2

The RC ciphers were designed by Ron Rivest for the RSA Data Security. RC stands for *Ron's Code* or *Rivest Cipher*. **RC2** was designed as a quick-fix replacement for DES, which is more secure. It is a block cipher with a variable key size that has a propriety algorithm. RC2 is a variable-key-length cipher. However, when using the Microsoft Base Cryptographic Provider, the key length is hard-coded to 40 bits. When using the Microsoft Enhanced Cryptographic Provider, the key length is 128 bits by default and can be in the range of 40 to 128 bits in 8-bit increments.

RC4 was developed by Ron Rivest in 1987. It is a variable-key-size stream cipher. The details of the algorithm have not been officially published, but are known. The algorithm is extremely easy to describe and program. Just like RC2, 40-bit RC4 is supported by the Microsoft Base Cryptographic provider, and the Enhanced provider allows keys in the range of 40 to 128 bits in 8-bit increments.

RC4 generates a pseudorandom stream of bits. As with any stream cipher, this key-stream can be used for encryption by XORing it with the plaintext using bit-wise operation. The decryption is performed in the same way. To generate the key-stream, the cipher makes use of a secret internal state which consists of two parts:

- (i) A permutation of all 256 possible bytes (denoted by S).
- (ii) Two 8-bit index-pointers (denoted by i and j).

Initially, the entries of S are set equal to the values from 0 through 255 in ascending order, that is, $S[0] = 0, S[1] = 1, \dots, S[255] = 255$. Based on a temporary vector, T (typically between 40 and 256-bit long), the contents of S are swapped. Because the only operation on S is a swap, the only effect is a permutation. S still contains all the numbers from 0 through 255. Once this has been completed, the stream of bits is generated using the *pseudo-random generation algorithm* (PRGA).

RC5 is a block cipher designed for speed. The block size, key size and the number of iterations are all variables. In particular, the key size can be as large as 2,048 bits.

All the encryption techniques discussed so far (DES, AES, IDEA, RC Ciphers) belong to the class of symmetric cryptography. We will next study the class of **Asymmetric Cryptographic Techniques**.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 2:

1. What is the difference between a message authentication code (MAC) and a one-way hash function? S
2. Consider RC4 with the internal state, S , and the two indices i and j . How many bits are used to store the internal state? Determine, in bits, how much information is represented by the state. S
3. List important design considerations for a stream cipher. S
4. What purpose do the S-boxes serve in DES? S
5. Suppose we launch a chosen-ciphertext attack on a *linear* block cipher of length 128 bits. This linear cipher satisfies: $E(P_1 \oplus P_2, k) = E(P_1, k) \oplus E(P_2, k)$, where \oplus represents the XOR function, P_1 and P_2 represent the plaintext, $E(P_i, k)$ represents the ciphertext using the encryption algorithm, E . How many chosen-ciphertexts would be needed to break the cryptosystem? D

If you have successfully solved the above problems,
you have mastered LO 2. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/623>



9.8 Asymmetric (Public-Key) Algorithms

Public-key Algorithms are asymmetric, that is, the key used to encrypt the message is different from the key used to decrypt the message. The encryption key, known as the **public key** is used to encrypt a message, but the message can only be decoded by the person that has the decryption key, known as the **private key**.

This type of algorithm has a number of advantages over traditional symmetric ciphers. It means that the recipient can make their public key widely available—anyone wanting to send them a message uses the algorithm and the recipient's public key to do so. An eavesdropper may have both the algorithm and the public key, but will still not be able to decrypt the message. Only the recipient, with their private key can decrypt the message.

LO 3



Underline the concept of public-key cryptography and implement the Rivest, Shamir and Adleman (RSA) algorithm, Pretty Good Privacy (PGP) and one-way Hashing.

DID YOU KNOW ? A disadvantage of public-key algorithms is that they are more computationally intensive than symmetric algorithms, and therefore encryption and decryption take longer. This may not be significant for a short text message, but certainly is for long messages or audio/video clips.

The **Public-Key Cryptography Standards** (PKCS) are specifications produced by RSA Laboratories in cooperation with secure systems developers worldwide for the purpose of accelerating the deployment of public-key cryptography. First published in 1991 as a result of meetings with a small group of early adopters of public-key technology, the PKCS documents have become widely referenced and implemented. Contributions from the PKCS series have become part of many formal and de facto standards, including ANSI X9 documents, PKIX, SET, S/MIME, and SSL.

The following section describes the RSA algorithm, followed by the Pretty Good Privacy (PGP) hybrid algorithm.

9.9 The RSA Algorithm

LO 3

RSA, named after its three creators—Rivest, Shamir and Adleman, was the first effective public-key algorithm, and for years has withstood intense scrutiny by cryptanalysts all over the world. Unlike symmetric key algorithms, where, as long as one presumes that an algorithm is not flawed, the security relies on having to try all possible keys, public-key algorithms rely on being computationally unfeasible to recover the private key from the public key.

RSA relies on the fact that it is easy to multiply two large prime numbers together, but extremely hard (i.e., time consuming) to factor them back from the result. Factoring a number means finding its prime factors, which are the prime numbers that need to be multiplied together in order to produce that number. For example,

$$10 = 2 \times 5$$

$$60 = 2 \times 2 \times 3 \times 5$$

$$2^{113} - 1 = 3391 \times 23279 \times 65993 \times 1868569 \times 1066818132868207$$

The Algorithm

Two very large prime numbers, normally of equal length, are randomly chosen and then multiplied together.

$$N = A \times B \quad (9.4)$$

$$T = (A - 1) \times (B - 1) \quad (9.5)$$

A third number is then also chosen randomly as the public key (E) such that it has no common factors (i.e., it is relatively prime) with T . The private key (D) is then

$$D = E^{-1} \bmod T \quad (9.6)$$

To encrypt a block of plaintext (M) into ciphertext (C):

$$C = M^E \bmod N \quad (9.7)$$

To decrypt

$$M = C^D \bmod N \quad (9.8)$$

Example 9.11 Consider the following implementation of the RSA algorithm.

$$1^{\text{st}} \text{ prime } (A) = 37$$

$$2^{\text{nd}} \text{ prime } (B) = 23$$

So,

$$N = 37 \times 23 = 851$$

$$T = (37 - 1) \times (23 - 1) = 36 \times 23 = 792$$

E must have no factors other than 1 in common with 792.

E (public key) could be 5.

$$D \text{ (private key)} = 5^{-1} \bmod 792 = 317$$

To encrypt a message (M) of the character ‘G’:

If G is represented as 7 (7th letter in alphabet), then $M = 7$.

$$C \text{ (ciphertext)} = 7^5 \bmod 851 = 638$$

To decrypt:

$$M = 638^{317} \bmod 851 = 7.$$

Example 9.12 The previous example was a toy-example to illustrate the working of RSA algorithm. Let us consider a slightly more realistic example of RSA algorithm.

$$1^{\text{st}} \text{ prime } (A) = 3283807$$

$$2^{\text{nd}} \text{ prime } (B) = 7365557$$

So,

$$N = 3283807 \times 7365557 = 24187067635499$$

$$T = (3283807 - 1) \times (7365557 - 1) = 24187056986136.$$

E must have no factors other than 1 in common with 24187056986136.

E (public key) could be 11828950082903.

Verify that $\gcd(24187056986136, 11828950082903) = 1$.

DID YOU
KNOW

D (private key) = $(11828950082903)^{-1} \bmod 792 = 6376223$.
In real-life, A and B contain more than 100 digits!

Prime Numbers

The RSA algorithm relies on large prime numbers. Do we actually have arbitrarily sized prime numbers? To answer this question, we first define the **Prime Counting Function**.

Definition 9.7 The **Prime Counting Function** $\pi(n)$ counts the number of primes that are less than or equal to n .

Example 9.13 Table 9.2 illustrates the prime counting function.

Table 9.2 The Prime Counting Function

n	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...
$\pi(n)$	1	2	2	3	3	4	4	4	4	5	5	6	6	6	...

From the table we note that we increase the value of $\pi(n)$ only when we encounter the next prime number. The function $\pi(n)$ is monotonically increasing.

Example 9.14 Do we have infinite number of primes? Let us assume that we have only finite number of prime numbers. Let these be p_1, p_2, \dots, p_n . Consider the number $m = p_1 p_2 \dots p_n + 1$. Because m is bigger than any prime, it must be a composite number. Hence it should be divisible by some prime number. However, it is not divisible by p_1 because we get the remainder '1' after dividing m by p_1 . Similarly, it is not divisible by any p_i , $i = 1, 2, \dots, n$. Thus we get a contradiction and hence our assumption was incorrect. Therefore, we have infinitely many primes.

Theorem 9.1 The Prime Density Theorem states that

$$\lim_{n \rightarrow \infty} \frac{\pi(n) \ln(n)}{n} = 1 \quad (9.9)$$

Typically, $\pi(n) \approx \frac{n}{\ln(n)}$ is a good approximation. This tells us that we do have lots of prime numbers.

There are two common methods of generating large prime numbers.

- (i) Construct provable primes
- (ii) Randomly choose large odd numbers and apply primality tests.

Example 9.15 Let us use the prime density theorem to estimate the number of 50-digit prime numbers. This is obtained by subtracting the total number of primes up to 49 digits from the total number of primes up to 50 digits.

Thus, the number of 50-digit prime numbers $\approx \pi(10^{50}) - \pi(10^{49})$

$$= \frac{10^{50}}{\ln(10^{50})} - \frac{10^{49}}{\ln(10^{49})} = 7.79 \times 10^{47}.$$

Out of curiosity, let us randomly pick a 50-digit odd number. What is the probability that it will be a prime number?

The fraction of 50-digit numbers that are prime $\approx (7.79 \times 10^{47} / 10^{50}) \approx 1/128$.

Since we are excluding all even numbers, the odds become twice as much (no pun intended!). Hence, the probability that a 50-digit odd number selected randomly turns out to be a prime number $\approx 1/64$.



Theorem 9.2 Fermat's Little Theorem states that suppose p is a prime and a is an integer that is not a multiple of p , then $a^{p-1} \equiv 1 \pmod{p}$.

Definition 9.8 The Primality Decision Problem: Given a positive integer n , decide whether n is a prime or not.

One of the direct outcomes of Fermat's little theorem is a simple primality test. If n is a positive integer and $a^{n-1} \neq 1 \pmod{n}$, for some number a , $1 < a < n - 1$, then n is composite. In order to describe the computational complexity of primality tests we require the following definitions.



Definition 9.9 A Turing Machine is an imaginary computing device that yields a primitive but sufficiently general computational model. It is typically represented by a **Finite State Machine**.

Definition 9.10 An algorithm is called **Polynomial Time** if its worst case running time function is polynomial in the input size. Any algorithm whose running time cannot be bounded by a polynomial is called **Super Polynomial Time**.

Definition 9.11 The **Complexity Class P** refers to the class of decision problems $D \subseteq \{0, 1\}$ that is solvable in polynomial time by a deterministic Turing machine.

 **Definition 9.12** The **Complexity Class NP** refers to the class of decision problems $D \subseteq \{0, 1\}$ for which a ‘yes’ answer can be verified in polynomial time, given some extra information called a certificate or witness. The **Complexity Class $\text{co-}NP$** refers to the class of decision problems $D \subseteq \{0, 1\}$ for which a ‘no’ answer can be verified in polynomial time, given some extra information called a certificate or witness.

The primality test algorithms can either be deterministic or probabilistic. Only a few deterministic primality testing functions are efficient, i.e., run in polynomial time. The probabilistic primality testing algorithms are much more efficient. Knowing efficient primality testing algorithms implies that the primality decision problem is in the complexity class P .

An important open question in complexity theory is whether $NP = P$ or $NP \neq P$? It is however widely believed that $NP \neq P$. This is also supported by our intuition that solving a problem is more involved than verifying the solution. Also, if $NP = P$, there would be no computationally secure cryptographic system in the mathematical sense.

Example 9.16 Let us use Fermat’s little theorem to check whether $n = 409$ is a prime. We use a probabilistic approach.

- (i) Randomly choose a base $a = 237$. Here $n = 409$.

$$a^{n-1} = 237^{408} \pmod{409} \equiv 1.$$

So, it passes the test for this randomly chosen base.

- (ii) Again, randomly choose another base $a = 356$.

$$a^{n-1} = 356^{408} \pmod{409} \equiv 1.$$

So, it again passes the test for this randomly chosen base.

- (iii) Again, randomly choose another base $a = 202$.

$$a^{n-1} = 202^{408} \pmod{409} \equiv 1.$$

So, it again passes the test for this randomly chosen base.

We are getting convinced (probabilistically)! Let us try one more time.

- (iv) Again, randomly choose another base $a = 105$.

$$a^{n-1} = 105^{408} \pmod{409} \equiv 1.$$

So, it again passes the test for this randomly chosen base.

So, probably 409 a prime.

Can you show by another method that indeed 409 is a prime number?



Let us see how RSA can be used for generating digital signatures. Suppose Bob requires Alice to sign a digital document, P . To do this, Alice will use her private key, D to encrypt P and send the digital signature, $s = D(P)$. Bob will then use the corresponding public key $E = D^{-1}$ to obtain $D^{-1}(s) = D^{-1}(D(P)) = P$ (original document). In case, $D^{-1}(s) \neq P$, Bob rejects the signature as invalid.

Example 9.17 Consider an RSA-based digital signature scheme. Suppose the RSA parameters are $A = 23$ and $B = 79$ and the keys are $D = 103$ and $E = 883$.

Let the digital message be $P = 1776$.

We first compute the digital signature, $s = D(P) = P^D \bmod N$.

Here,

$$N = A \times B = 1817.$$

Thus,

$$s = 1776^{103} \pmod{1817} = 1790.$$

To verify the digital signature, we must use the public key, E .

$$D^{-1}(s) = s^E \bmod N = 1790^{883} \pmod{1817} = 1776.$$

Thus, the signature is verified.

Security of RSA

The security of RSA algorithm depends on the ability of the hacker to factorise numbers. New, faster and better methods for factoring numbers are constantly being devised. The current best for long numbers is the *Number Field Sieve*. Prime Numbers of a length that was unimaginable a mere decade ago are now factored easily. Obviously the longer number is, the harder it is to factor, and so the better the security of RSA. As theory and computers improve, larger and larger keys will have to be used. The disadvantage in using extremely long keys is the computational overhead involved in encryption/decryption. This will only become a problem if a new factoring technique emerges that requires keys of such lengths to be used that the necessary key length increases much faster than the increasing average speed of computers utilising the RSA algorithm. The security of RSA cryptosystems may get threatened in the following two ways.

- (i) If a polynomial time algorithm for factoring primes gets discovered, or
- (ii) If a new revolution in computer hardware/distributed computing/quantum computing takes place.

DID YOU KNOW ? In 1997, a specific assessment of the security of 512-bit RSA keys shows that one may be factored for less than \$1,000,000 in cost and eight months of effort. It is therefore believed that 512-bit keys provide insufficient security for anything other than short-term needs. RSA Laboratories currently recommends key sizes of 768 bits for personal use, 1024 bits for corporate use, and 2048 bits for extremely valuable keys like the root-key pair used by a certifying authority. Security can be increased by changing a user's keys regularly and it is typical for a user's key to expire after two years (the opportunity to change keys also allows for a longer length key to be chosen).

Even without using huge keys RSA is about 1000 times slower to encrypt/decrypt than DES, this has resulted in it not being widely used as a stand-alone cryptography system. However, it is used in many hybrid cryptosystems such as PGP. The basic principle of hybrid systems is to encrypt plaintext with a symmetric algorithm (usually DES or IDEA); the symmetric algorithm's key is then itself encrypted with a public-key algorithm such as RSA. The RSA-encrypted key and symmetric algorithm-encrypted message are then sent to the recipient, who uses his private RSA key to decrypt the symmetric algorithm's key, and then that key to decrypt the message. This is considerably faster than using RSA throughout, and allows a different symmetric key to be used each time, considerably enhancing the security of the symmetric algorithm.



INDUSTRIAL RELEVANCE

RSA's future security relies solely on advances in factoring techniques. Barring an astronomical increase in the efficiency of factoring techniques, or available computing power, the 2048-bit key will ensure very secure protection into the foreseeable future. For instance an Intel Paragon, which can achieve 50,000 mips (million operations per second), would take a million years to factor a 2048-bit key using current techniques. The ANSI X9.31 standard specifies a signature mechanism based on an RSA signature algorithm. ISO 11166 is a standard for banking key management. Part 2 of this standard prescribes the RSA algorithm for both encryption and digital signatures.

LO 3

9.10 Pretty Good Privacy (PGP)

Pretty Good Privacy (PGP) is a hybrid cryptosystem that was created by Phil Zimmerman and released onto the Internet as a freeware program in 1991. PGP is not a new algorithm in its own right, but rather a series of other algorithms that are performed along with a sophisticated protocol. PGP's intended use was for e-mail security, but there is no reason why the basic principles behind it could not be applied to any type of transmission.

PGP and its source code is freely available on the Internet. This means that since its creation PGP has been subjected to an enormous amount of scrutiny by cryptanalysts, who have yet to find an exploitable fault in it.

PGP has four main modules: a symmetric cipher—IDEA for message encryption; a public-key algorithm—RSA to encrypt the IDEA key and hash values; a one-way hash function—MD5 for signing and a random number generator.

The fact that the body of the message is encrypted with a symmetric algorithm (IDEA) means that PGP generated e-mails are a lot faster to encrypt and decrypt than using simple RSA. The key for the IDEA module is randomly generated each time as a one-off session key, this makes PGP very secure, as even if one message is cracked, all previous and subsequent messages would remain secure. This session key is then encrypted with the public key of the recipient using RSA. Given that keys up to 2048 bits long can be used, this is extremely secure. MD5 can be used to produce a hash of the message, which can then be signed by the sender's private key. Another feature of PGP's security, is that the user's private key is encrypted using a hashed pass-phrase rather than simply a password, making the private key extremely resistant to copying even with access to the user's computer.

Generating true random numbers on a computer is notoriously hard, PGP tries to achieve randomness by making use of the keyboard latency when the user is typing. This means that the program measures the gap of time between each key-press. While at first this may seem to be distinctly non-random, it is actually fairly effective—people take longer to hit some keys than others, pause for thought, make mistakes and vary their overall typing speed on all sorts of factors such as knowledge of the subject and tiredness. These measurements are not actually used directly, but used to seed a pseudo-random number generator. There are other ways of generating random numbers, but to be much better than this gets very complex.

DID YOU
KNOW ?

PGP uses a very clever, but complex, protocol for key management. Each user generates and distributes their public key. If James is happy that a person's public key belongs to who it claims to belong to, then he can sign that person's public key, showing others they believe this

to be true, and James's program will then accept messages from that person as valid. The user can allocate levels of trust to other users. For instance, James may decide that he completely trusts Earl to sign other peoples' keys, in effect saying "his word is good enough for me". This means that if Rachel, who has had her key signed by Earl, wants to communicate with James, she sends James her signed key. The program recognises Earl's signature, has been told that Earl can be trusted to sign keys and so accepts Rachel's key as valid. In effect Earl has introduced Rachel to James.

PGP allows many levels of trust to be assigned to people, and this is best illustrated in Fig. 9.4.

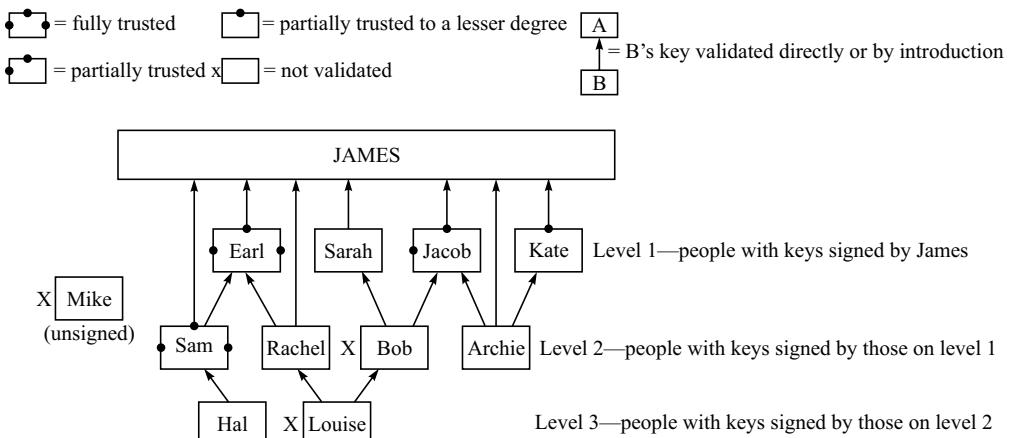


Fig. 9.4 An example of a PGP user web.

The explanations are as follows.

1st line

James has signed the keys of Earl, Sarah, Jacob and Kate. James completely trusts Earl to sign other peoples' keys, does not trust Sarah at all, and partially trusts Jacob and Kate (he trusts Jacob more than Kate).

2nd line

Although James has not signed Sam's key he still trusts Sam to sign other peoples' keys, maybe on Bob's word so or due to them actually meeting. Because Earl has signed Rachel's key, Rachel is validated (but not trusted to sign keys). Even though Bob's key is signed by Sarah and Jacob, because Sarah is not trusted and Jacob only partially trusted, Bob is not validated. Two partially trusted people, Jacob and Kate, have signed Archie's key, therefore Archie is validated.

3rd line

Sam, who is fully trusted, has signed Hal's key, therefore Hal is validated. Louise's key has been signed by Rachel and Bob, neither of whom is trusted, therefore Louise is not validated.

Odd one out

Mike's key has not been signed by anyone in James' group, maybe James found it on the Internet and does not know whether it is genuine or not.

PGP never prevents the user from sending or receiving e-mail, it does however warn the user if a key is not validated, and the decision is then up to the user as to whether to heed the warning or not.

Key revocation

If a user's private key is compromised then they can send out a key revocation certificate. Unfortunately this does not guarantee that everyone with that user's public key will receive it, as keys are often swapped in a disorganised manner. Additionally, if the user no longer has the private key then they cannot issue a certificate, as the key is required to sign it.

Security of PGP

'A chain is only as strong as its weakest link' is the saying, and it holds true for PGP. If the user chooses a 40-bit RSA key to encrypt his session keys and never validates any users, then PGP will not be very secure. If however a 2048-bit RSA key is chosen and the user is reasonably vigilant, then PGP is the closest thing to military-grade encryption the public can hope to get their hands on.

The Ex-Deputy Director of the NSA was quoted as saying:

'If all the personal computers in the world, an estimated 260 million, were put to work on a single PGP-encrypted message, it would still take an estimated 12 million times the age of the universe, on average, to break a single message'.

A disadvantage of public-key cryptography is that anyone can send you a message using your public key and it is then necessary to prove that this message came from whom it claims to have been sent by. A message encrypted by someone's private key, can be decrypted by anyone with their public key. This means that if the sender encrypted a message with his private key, and then encrypted the resulting ciphertext with the recipient's public key, the recipient would be able to decrypt the message with first their private key, and then the sender's public key, thus recovering the message and proving it came from the correct sender.

This process is very time-consuming, and therefore rarely used. A much more common method of digitally signing a message is using a method called one-way hashing.

9.11 One-way Hashing

LO 3

A **One-Way Hash Function** is a mathematical function that takes a message string of any length (pre-string) and returns a smaller fixed-length string (hash value). These functions are designed in such a way that not only is it very difficult to deduce the message from its hashed version, but also given that all hashes are of a certain length, it is extremely hard to find two messages that hash to the same value. In fact to find two messages with the same hash from a 128-bit hash function, 2^{64} hashes would have to be tried. In other words, the hash value of a file is a small unique 'fingerprint'. Even a slight change in an input string should cause the hash value to change drastically. Even if 1 bit is flipped in the input string, at least half of the bits in the hash value will flip as a result. We have studied this earlier as the avalanche effect.

H = hash value, f = hash function, M = original message/pre-string

$$H = f(M) \quad (9.10)$$

If you know M then H is easy to compute. However, knowing H and f , it is not easy to compute M , and is hopefully computationally unfeasible.

As long as there is a low risk of collision (i.e., 2 messages hashing to the same value), and the hash is very hard to reverse, then a one-way hash function proves extremely useful for a number of aspects of cryptography.

If you one-way hash a message, the result will be a much shorter but still unique (at least statistically) number. This can be used as proof of ownership of a message without having to reveal the contents of the actual message. For instance, rather than keeping a database of copyrighted documents, if just the hash values of each document were stored, then not only would this save a lot of space, but it would also provide a great

deal of security. If copyright needs to be proved after that, the owner could produce the original document and prove it hashes to that value.

Hash-functions can also be used to prove that no changes have been made to a file, as adding even one character to a file would completely change its hash value.

By far the most common use of hash functions is to digitally sign messages. The sender performs a one-way hash on the plaintext message, encrypts it with his private key and then encrypts both with the recipient's public key and sends in the usual way. On decrypting the ciphertext, the recipient can use the sender's public key to decrypt the hash value, he can then perform a one-way hash himself on the plaintext message and check this with the one he has received. If the hash values are identical, the recipient knows not only that the message came from the correct sender, as it used their private key to encrypt the hash, but also that the plaintext message is completely authentic as it hashes to the same value.

The above method is greatly preferable to encrypting the whole message with a private key, as the hash of a message will normally be considerably smaller than the message itself. This means that it will not significantly slow down the decryption process in the same way that decrypting the entire message with the sender's public key, and then decrypting it again with the recipient's private key would. The PGP system uses the MD5 hash function for precisely this purpose.

INDUSTRIAL RELEVANCE



The Microsoft cryptographic providers support three hash algorithms: MD4, MD5 and **Secure Hash Algorithm** (SHA). Both MD4 and MD5 were invented by Ron Rivest. MD stands for Message Digest. Both algorithms produce 128-bit hash values. MD5 is an improved version of MD4. SHA was designed by NIST and NSA. SHA produces 160-bit hash values, longer than MD4 and MD5. SHA is generally considered more secure than other algorithms and is the recommended hash algorithm. The ANSI public-key standards, X9.30-1 and X9.30-2, specify DSA and SHA for the financial services industry. The hash algorithm specified in the original standard FIPS 180 is the SHA.

Example 9.18 Consider a room with N people. For what value of N do we expect to find at least one pair of persons with the same date of birth? Alternately, how large must N be such that two or more people in the room share their birthdays with probability greater than 0.5?

Let us solve the complementary problem, and then we will subtract the probability from 1. Person 1 can have his birthday on any one of the 365 days. If Person 2 must have a different date of birth, then he can have his birthday on the remaining 364 days. Similarly, Person 3 can have a birthday on the remaining 363 days, and so on. Assuming all birthdays are equally likely, the probability that two or more people in the room share their birthdays is

$$P = 1 - \left(\frac{365}{365} \right) \left(\frac{364}{365} \right) \left(\frac{363}{365} \right) \dots \left(\frac{365 - N + 1}{365} \right)$$

Setting $P = 0.5$, we get $N = 23$. Thus, we need only 23 people in the room and expect to find two or more with the same birthday! This is called the **Birthday paradox**.



Intuition Consider another perspective. With N people in the room we have $\binom{N}{2} = \frac{N(N-1)}{2} \approx N^2$ comparisons. Setting $N^2 = 365$ we get $N = \sqrt{365} \approx 19$. Hence, the paradox is not too bad!

Let us link the birthday paradox to hash functions. Suppose a hash function produces an output that is N bits long. There will be 2^N possible hash functions of that length. Let us assume that all outputs are equally likely. Then, from the birthday paradox, if we hash approximately $\sqrt{2^N} = 2^{N/2}$ different inputs, we can expect to get a collision. Thus, a brute force attack on an N -bit long hash function would need $2^{N/2}$ attempts. Comparing it to a brute force attack on a symmetric-key cipher of key length N , one would need $\frac{1}{2} \times 2^N = 2^{N-1}$ attempts. Consequently, an N -bit long hash function is weaker, and needs to be almost twice as long as the key of a symmetric-key cipher for equivalent levels of security against brute force attack.

9.12 Other Techniques

LO 3

One Time Pads

The one-time pad was invented by Major Joseph Mauborgne and Gilbert Bernam in 1917, and is an unconditionally secure (i.e. unbreakable) algorithm. The theory behind a one-time pad is simple. The pad is a non-repeating random string of letters. Each letter on the pad is used once only to encrypt one corresponding plaintext character. After use, the pad must *never* be re-used. As long as the pad remains secure, so is the message. This is because a random key added to a non-random message produces completely random ciphertext, and there is absolutely no amount of analysis or computation that can alter that. If both pads are destroyed then the original message will never be recovered. There are two major drawbacks: Firstly, it is extremely hard to generate truly random numbers, and a pad that has even a couple of non-random properties is theoretically breakable. Secondly, because the pad can never be reused no matter how large it is, the length of the pad must be the same as the length of the message—fine for text, but virtually impossible for a video.

Steganography

Steganography is not actually a method of encrypting messages, but hiding them within something else to enable them to pass undetected. Traditionally, this was achieved with invisible ink, microfilm or taking the first letter from each word of a message. This is now achieved by hiding the message within a graphics or sound file. For instance in a 256-greyscale image, if the least significant bit of each byte is replaced with a bit from the message then the result will be indistinguishable to the human eye. An eavesdropper will not even realise a message is being sent. This is not cryptography however, and although it would fool a human, a computer would be able to detect this very quickly and reproduce the original message.

Secure Mail and S/MIME

DID YOU KNOW ? Secure Multipurpose Internet Mail Extensions (S/MIME) is a *de facto* standard developed by RSA Data Security, Inc., for sending secure mail based on public-key cryptography. MIME is the industry standard format for electronic mail, which defines the structure of the message's body. S/MIME-supporting e-mail applications add digital signatures and encryption capabilities to that format to ensure message integrity, data origin authentication and confidentiality of electronic mail.

When a signed message is sent, a detached signature in the PKCS #7 format is sent along with the message as an attachment. The signature attachment contains the hash of the original message signed with the sender's private key, as well as the signer certificate. S/MIME also supports messages that are first signed with the sender's private key and then enveloped using the recipient's public keys.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 3:

1. Use the prime numbers 29 and 61 to generate keys using the RSA algorithm.
 - (i) Show that $E = 11$ is a valid public key.
 - (ii) Represent the letters 'RSA' in ASCII ($R = 82$, $S = 83$, $A = 65$) and encode them using the key generated above.
2. Use RSA to perform encryption and decryption with: $A = 17$, $B = 31$, $E = 7$ and message $M = 2$.
3. In an RSA-based cryptosystem, the public key of a given user is $E = 31$, $N = 3599$. Can you determine the private key?
4. What is the probability that two random numbers are relatively prime?
5. Consider the messages in the form of a sequence of decimal numbers, $M = (m_1, m_2, \dots, m_i)$

and the corresponding hash value h calculated as $\sum_{i=1}^K m_i \bmod n$ for some value of n and K .

Is this a good hash function?

If you have successfully solved the above problems,
you have mastered LO 3. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/624>



9.13 Elliptic Curve Cryptography

LO 4



Understand and solve problems related to the Diffie-Hellman protocol, elliptic curve cryptography, quantum cryptography, biometric encryption and chaos-based cryptography.

Most public key cryptosystems get their security from the assumed difficulty of inverting a one-way function. However, inverting one-way functions are not equally difficult in all algebraic structures. **Elliptic Curve Cryptography** (ECC) has become important mainly because groups have been found in which sub-exponential algorithms inverting the discrete exponentiation function are not known to exist. Thus one has to use the standard exponential time algorithms to break the security of the conventional public key cryptosystems. The basic advantage of elliptic curve cryptosystems is that they are equally secure with smaller key sizes than their conventional counterparts (e.g., RSA).

Let \mathbf{Z}_p be the set of integers $\{0, 1, 2, \dots, p-1\}$ where p is a prime number. Let us define an elliptic curve over \mathbf{Z}_p as follows.

$$y^2 \equiv x^3 + ax + b \pmod{p} \quad (9.11)$$

with $a, b \in \mathbf{Z}_p$ and $4a^3 + 27b^2 \neq 0 \pmod{p}$. For any given a and b in \mathbf{Z}_p , the above equation has a pair of solution x, y in \mathbf{Z}_p which can be expressed as

$$\begin{aligned} E(\mathbf{Z}_p) = & \{(x, y) \mid x, y \in \mathbf{Z}_p \text{ and} \\ & y^2 \equiv x^3 + ax + b \pmod{p} \text{ and} \\ & 4a^3 + 27b^2 \neq 0 \pmod{p}\} \end{aligned} \quad (9.12)$$

The resulting set $E(\mathbf{Z}_p)$ consists of all $(x, y) \in \mathbf{Z}_p^2$ that satisfy (9.12). In addition to the points lying on the elliptic curve, one also considers a point \mathbf{O} at infinity. Some properties of the points on the elliptic curve are listed below.

- (i) $\mathbf{P} + \mathbf{O} = \mathbf{O} + \mathbf{P}$ for all $\mathbf{P} \in E(\mathbf{Z}_p)$.
- (ii) If $\mathbf{P} = (x, y) \in E(\mathbf{Z}_p)$, then $(x, y) + (x, -y) = \mathbf{O}$. The point $(x, -y)$ is also called $-\mathbf{P}$. Note that $-\mathbf{P}$ will always be a point on the elliptic curve.
- (iii) Let $\mathbf{P} = (x_1, y_1) \in E(\mathbf{Z}_p)$ and $\mathbf{Q} = (x_2, y_2) \in E(\mathbf{Z}_p)$ with $\mathbf{P} \neq -\mathbf{Q}$, then $\mathbf{P} + \mathbf{Q} = (x_3, y_3)$ where

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \\ \lambda &= \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } \mathbf{P} \neq \mathbf{Q} \\ \frac{3x_1^2 + a}{2y_1} & \text{if } \mathbf{P} = \mathbf{Q} \end{cases} \end{aligned} \quad (9.13)$$

Thus it is fairly easy to calculate $2\mathbf{P}, 3\mathbf{P}, \dots, k\mathbf{P}$. Self-addition can also be carried out efficiently using geometric techniques. The Elliptic Curve Cryptography relies on the difficulty of finding k given $\mathbf{Q} = k\mathbf{P}$. This property will be used in the next section when we discuss the Diffie-Hellman protocol based on Elliptic Curve.

Here are some properties of elliptic curve addition.

Commutativity: $\mathbf{P} + \mathbf{Q} = \mathbf{Q} + \mathbf{P}$.

Associativity: $(\mathbf{P} + \mathbf{Q}) + \mathbf{R} = \mathbf{P} + (\mathbf{Q} + \mathbf{R})$.

Additive identity: $\mathbf{P} + \mathbf{O} = \mathbf{P}$, where \mathbf{O} is a point at infinity.

Additive inverse: $\mathbf{P} + (-\mathbf{P}) = \mathbf{O}$, and $-\mathbf{O} = \mathbf{O}$.

Example 9.19 Let $p = 23$, $a = 1$ and $b = 1$. Thus, the elliptic curve can be represented as $y^2 \equiv x^3 + x + 1$ defined over \mathbf{Z}_{23} . One can verify that indeed $4a^3 + 27b^2 \neq 0 \pmod{p}$. The valid points on the elliptic curve are \mathbf{O} and the following.

(0, 1) (0, 22) (1, 7) (1, 16) (3, 10) (3, 13) (4, 0) (5, 4) (5, 19) (6, 4) (6, 19) (7, 11) (7, 12) (9, 7) (9, 16) (11, 3) (11, 20) (12, 4) (12, 19) (13, 7) (13, 16) (17, 3) (17, 20) (18, 3) (18, 20) (19, 5) (19, 18)

Let $\mathbf{P} = (3, 10)$ and $\mathbf{Q} = (9, 7)$. In order to compute $\mathbf{P} + \mathbf{Q}$, we have to first determine λ using (9.13).

$$\lambda = \frac{7-10}{9-3} = \frac{-3}{6} = \frac{-1}{2} = 11 \in \mathbf{Z}_{23}. \text{ Consequently,}$$

$$\begin{aligned}x_3 &= 11^2 - 3 - 9 \equiv 17 \pmod{23} \\y_3 &= 11(3 - (-6)) - 10 \equiv 20 \pmod{23}\end{aligned}$$

Thus, $\mathbf{P} + \mathbf{Q} = (17, 20) \in E(\mathbf{Z}_{23})$.

Next, let us determine $\mathbf{P} + \mathbf{P} = 2\mathbf{P}$. Again, from (9.13),

$$\lambda = \frac{3(3^2) + 1}{20} = \frac{5}{20} = \frac{1}{4} = 6 \in \mathbf{Z}_{23}.$$

$$\begin{aligned}x_3 &= 6^2 - 3 - 3 \equiv 7 \pmod{23} \\y_3 &= 11(3 - 7) - 10 \equiv 12 \pmod{23}\end{aligned}$$

Thus, $2\mathbf{P} = (7, 12) \in E(\mathbf{Z}_{23})$.

Definition 9.13 If $p > 3$ is a prime number and E is an elliptic curve mod p , and $\mathbf{P}, \mathbf{Q} \in E$, the **Elliptic Curve Discrete Logarithm problem** is to find a positive integer m such that $\mathbf{Q} = m\mathbf{P}$, if such an integer exists.

Example 9.20 Let us look at the elliptic curve discrete logarithm problem. Consider the Elliptic curve with $p = 23$, $a = 9$ and $b = 17$. Thus, the elliptic curve can be represented as $y^2 \equiv x^3 + 9x + 17$ defined over \mathbf{Z}_{23} . We will try to answer the following question: What is the discrete log k of $\mathbf{Q} = (4, 5)$ to the base $\mathbf{P} = (16, 5)$? That is, how many times should \mathbf{P} be added to itself to obtain \mathbf{Q} ? The brute force method is to calculate $2\mathbf{P}$, $3\mathbf{P}$, ... until $k\mathbf{P}$ matches with \mathbf{Q} . For example,

$\mathbf{P} = (16, 5)$, $2\mathbf{P} = (20, 20)$, $3\mathbf{P} = (14, 14)$, $4\mathbf{P} = (19, 20)$, $5\mathbf{P} = (13, 10)$, $6\mathbf{P} = (7, 3)$, $7\mathbf{P} = (8, 7)$, $8\mathbf{P} = (12, 17)$, $9\mathbf{P} = (4, 5)$. Thus, $k = 9$.

In the real world k is large, and finding k is computationally very expensive. As stated before, the Elliptic Curve Cryptography relies on the difficulty of finding k given $\mathbf{Q} = k\mathbf{P}$.

Example 9.21 Let us look at a real-world problem. Consider the elliptic curve given by

$$E: y^2 \equiv x^3 + ax + b \pmod{p}, \text{ where}$$

$$p = 564538252084441556247016902735257$$

$$a = 321094768129147601892514872825668$$

$$b = 430782315140218274262276694323197$$

Let a point on the curve be denoted by

$$\mathbf{P} = (97339010987059066523156133908935, 149670372846169285760682371978898) \in E.$$

Let, k be 281183840311601949668207954530384. We can use the elliptic curve to add \mathbf{P} to itself k times. Let $\mathbf{Q} = k\mathbf{P}$. Then,

$$\mathbf{Q} = (44646769697405861057630861884284, 522968098895785888047540374779097).$$

9.14 Diffie-Hellman Key Agreement Protocol

LO 4

The Diffie-Hellman key agreement protocol (also called exponential key agreement) was developed by Diffie and Hellman in 1976 and published in the ground-breaking paper ‘New Directions in Cryptography’ in *IEEE Transactions on Information Theory*. The protocol allows two users to exchange a secret key over an insecure medium without any prior secrets.

The protocol has two system parameters p and g . They are both public and may be used by all the users in a system. Parameter p is a prime number and parameter g (usually called a generator) is an integer less than p , with the following property: for every number n between 1 and $p-1$ inclusive, there is a power k of g such that

$$n = g^k \bmod p \quad (9.14)$$

Suppose Alice and Bob want to agree on a shared secret key using the Diffie-Hellman key agreement protocol. They carry out the following steps.

- (i) Alice generates a random private value a and Bob generates a random private value b . Both a and b are drawn from the set of integers.
- (ii) Then they derive their public values using parameters p and g and their private values. Alice's public value is $g^a \bmod p$ and Bob's public value is $g^b \bmod p$.
- (iii) They then exchange their public values.
- (iv) Finally, Alice computes $g^{ab} = (g^b)^a \bmod p$, and Bob computes $g^{ba} = (g^a)^b \bmod p$. Since $g^{ab} = g^{ba} = k$, Alice and Bob now have a shared secret key k .

Example 9.22 Let us implement the Diffie-Hellman protocol using Elliptic Curve Cryptography.

The steps are as follows.

- (i) Alice and Bob mutually agree on a number P .
- (ii) Alice chooses a random private key k_A . She then computes $k_A P$ using the elliptic curve and publishes it.
- (iii) Bob chooses a random private key k_B . He then computes $k_B P$ using the same elliptic curve and publishes it.
- (iv) Alice takes $k_B P$ and uses the elliptic curve to calculate $k_A(k_B P)$.
- (v) Bob takes $k_A P$ and uses the elliptic curve to calculate $k_B(k_A P)$.
- (vi) Since $k_A(k_B P) = k_B(k_A P)$, Alice and Bob now share a common key.

This protocol is delineated in Fig. 9.5.

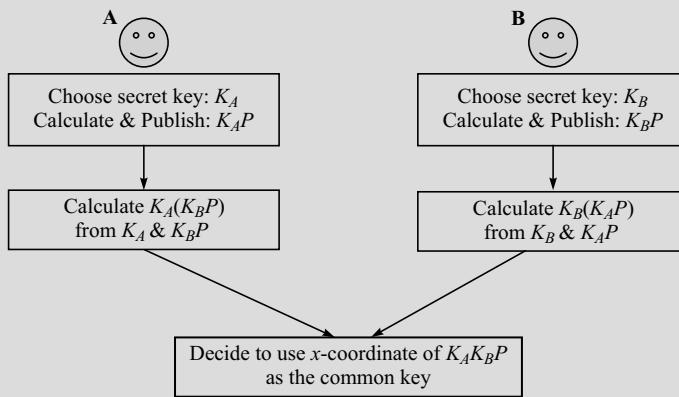


Fig. 9.5 Illustration of the Diffie-Hellman key exchange protocol using elliptic curve.

The protocol depends on the discrete logarithm problem for its security. It assumes that it is computationally infeasible to calculate the shared secret key $k = g^{ab} \bmod p$ given the two public values $g^a \bmod p$ and $g^b \bmod p$ when the prime p is sufficiently large.

The Diffie-Hellman key exchange is vulnerable to a **Man-in-the-middle Attack**. In this attack, an opponent Carol intercepts Alice's public value and sends her own public value to Bob. When Bob transmits his public value, Carol substitutes it with her own and sends it to Alice. Carol and Alice thus agree on one shared key and Carol and Bob agree on another. After this exchange, Carol simply decrypts any messages sent out by Alice or Bob, and then reads and possibly modifies them before re-encrypting with the appropriate key and transmitting them to the other party. This vulnerability is present because Diffie-Hellman key exchange does not authenticate the participants. Possible solutions include the use of digital signatures and other protocol variants.

INDUSTRIAL RELEVANCE



The ANSI X9.42 standard specifies several variations of unauthenticated Diffie-Hellman key agreement for sharing secret keys. ISO 11166 is a standard for banking key management. It specifies asymmetric techniques, including the Diffie-Hellman protocol, for distributing keys for symmetric algorithms.

9.15 Secure Communication Using Chaos Functions

LO 4

Chaos functions have also been used for secure communications and cryptographic applications. The implication of a chaos function here is an iterative difference equation that exhibits chaotic behaviour. If we observe the fact that cryptography has more to do with unpredictability rather than randomness, chaos functions are a good choice because of their property of *unpredictability*. If a hacker intercepts part of the sequence, he will have no information on how to predict what comes next. The unpredictability of chaos functions makes them a good choice for generating the keys for symmetric cryptography.

Example 9.23 Consider the difference equation

$$x_{n+1} = ax_n(1 - x_n) \quad (9.15)$$

For $a = 4$, this function behaves like a chaos function, i.e., (i) the values obtained by successive iterations are unpredictable and (ii) the function is extremely sensitive to the initial condition, x_0 . For any given initial condition, this function will generate values of x_n between 0 and 1 for each iteration. These values are good candidates for *key* generation. In single-key cryptography, a key is used for enciphering the message. This key is usually a pseudo noise (PN) sequence. The message can be simply XORed with the key in order to scramble it. Since x_n takes positive values that are always less than unity, the binary equivalent of these fractions can serve as keys. Thus, one of the ways of generating keys from these random, unpredictable decimal numbers is to directly use their binary representation. The lengths of these binary sequences will be limited only by the accuracy of the decimal numbers, and hence very long binary keys can be generated. The recipient must know the initial condition in order to generate the keys for decryption.

For application in single key cryptography the following two factors need to be decided.

- (i) The start value for the iterations (x_0), and
- (ii) The number for decimal places of the mantissa that are to be supported by the calculating machine (to avoid round off error).

For single-key cryptography, the chaos values obtained after some number of iteration are converted to binary fractions whose first 64 bits are taken to generate PN sequences. These initial iterations would make it still more difficult for the hacker to guess the initial condition. The starting value should be taken between 0 and 1. A good choice of the starting value can improve the performance slightly.

The secrecy of the starting number, x_0 , is the key to the success of this algorithm. Since chaos functions are extremely sensitive to even errors of 10^{-30} in the starting number (x_0), it means that we can have 10^{30} unique starting combinations. Therefore, a hacker who knows the chaos function and the encryption algorithm has to try out 10^{30} different start combinations. In the DES algorithm the hacker had to try out approximately 10^{19} different key values.

Chaos based algorithms require a high computational overhead to generate the chaos values as well as high computational speeds. Hence it might not be suitable for bulk data encryption.

9.16 Quantum Cryptography

LO 4

DID YOU KNOW ? While classical cryptography employs various mathematical techniques to restrict eavesdroppers from learning the contents of encrypted messages, in **Quantum Cryptography** the information is protected by the laws of physics. In classical cryptography an absolute security of information cannot be guaranteed. Quantum cryptography provides means for two parties to exchange an enciphering key over a private channel with *complete* security of communication. The Heisenberg Uncertainty Principle and quantum entanglement is exploited in such a system of secure communication.

A quantum cryptosystem can be explained as follows. Suppose the cryptosystem includes a transmitter and a receiver. A sender uses the transmitter to send photons in one of four polarisations: 0° , 45° , 90° , or 135° . The intended recipient at the other end uses the receiver to measure the polarisation. According to the laws of quantum mechanics, the receiver can distinguish between rectilinear polarisations (0° and 90°), or the diagonal polarisations (45° and 135°). However, it can never distinguish both types of polarisations. The key distribution requires several steps. The sender sends photons with one of the four polarisations, which are chosen at random. For each incoming photon, the receiver chooses at random the type of measurement: either the rectilinear type or the diagonal type. The receiver records the results of the measurements, but keeps them secret. Subsequently, the receiver publicly announces the type of measurement (but not the results) and the sender tells the receiver which measurements were of the correct type. The two parties (the sender and the receiver) keep all cases in which the receiver measurements were of the correct type. These cases are then translated into bits (1's and 0's) and thereby become the key.

An eavesdropper is bound to introduce errors to this transmission because he/she does not know in advance the type of polarisation of each photon. Quantum mechanics does not allow him/her to acquire sharp values of two non-commuting observables (here rectilinear and diagonal polarisations). The two legitimate users of the quantum channel test for eavesdropping by revealing a random subset of the key bits and checking (in public) the error rate. Although they cannot prevent eavesdropping, they can always detect the presence of an eavesdropper. This is because any, however subtle and sophisticated, effort to tap the channel will be detected. Whenever they are not happy with the security of the channel they can try to set up the key distribution again. Thus, quantum cryptography allows the users to communicate over the channel whenever it is perfectly safe.

Example 9.24 Suppose A and B wish to exchange a key over an insecure channel using a quantum cryptographic technique. Let the rectilinear basis be denoted by + and the diagonal basis by \times . The rectilinear polarisations (0° and 90°) are denoted by (\uparrow and \rightarrow) while the diagonal polarisations (45° and 135°) are denoted by (/ and \). The steps are as follows.

A's choice of random bits	1	0	0	1	0	1	1	0
A's choice of random basis	+	\times	\times	\times	+	\times	+	+
Polarisation sent by A	\rightarrow	/	/	\	\uparrow	\	\rightarrow	\uparrow
B's choice of random basis	+	+	\times	+	\times	\times	\times	+
Polarisation measured by B	\rightarrow	\rightarrow	/	\rightarrow	/	\	/	\uparrow
Discussion of basis in public								
Shared secret key	1		0			1		0

Now, A and B will disclose a subset of the key and compare whether all the bits are identical. Suppose they compare the second and fourth bits then, in this example, all the bits match, and hence A and B can use the remaining two bits (the first and third) as the secret key. In the presence of an eavesdropper, some of the bits in the key obtained by B will not match with those of A.

9.17 Biometric Encryption

LO 4

Conventional cryptography uses encryption keys, which are just bit strings long enough. These keys, either ‘public’ or ‘private’, are an essential part of any cryptosystem. Typically, a user of the cryptosystem cannot memorise such long random keys. Hence, the keys are generated, after several steps, from a password or a PIN that can be memorised. The password management is the weakest point of any cryptosystem, as the password can be guessed, found with a brute force search, or stolen by an attacker. In this section we will study a cryptosystem based on **Biometrics**.

Definition 9.14 **Biometrics** refers to automatic systems that use measurable, physical or physiological characteristics or behavioural traits to recognise the identity, or verify/authenticate the claimed identity of an individual.



Intuition Some examples of biometric characteristics that may be used for automated recognition are fingerprints, iris, face, hand or finger geometry, retina, voice, signature, keystroke dynamics, body odour and DNA. Can some of these biometric characteristics be directly used as a cryptographic key?

Unfortunately, the answer is negative: biometric images or templates are variable by nature, i.e., each new biometric sample is always different. We recall that the conventional cryptosystems do not tolerate a single bit error in the key.

A biometric system always produces a ‘Yes’/‘No’ response, which is essentially one bit of information. Therefore, an obvious role of biometrics in the conventional cryptosystem is just password management. Upon receiving ‘Yes’ response, the system unlocks a password or a key. The key must be stored in a secure location (so called ‘trusted’ device). This scheme is still prone to the security vulnerabilities since the biometric system and the application are connected via one bit only. Biometric templates or images stored in

a database can be encrypted by conventional cryptographic means. This would improve the level of system security, since an attacker must gain the access to the encryption keys first. However, most privacy issues associated with a large database remain, since the keys and, therefore, the biometric data, are controlled by a custodian.

It is obvious that because of its variability, the biometric image or template itself cannot serve as a cryptographic key. However, the amount of information contained in a biometric image is quite large: for example, a typical image of 300×300 pixel size, encoded with eight bits per pixel has $300 \times 300 \times 8 = 720,000$ bits of information. Of course, there is a fair amount of redundancy in this data. Let us ask the following questions:

- (i) Is it possible to *consistently* extract a relatively small number of bits, say 128, out of these 720,000 bits?
- (ii) Is it possible to bind a 128-bit key to the biometric information, so that the key could be consistently regenerated?

While the answer to the first question is tricky, the second question has given rise to the new area of research, called Biometric Encryption.

Definition 9.15 Biometric Encryption is a process that securely binds a PIN or a cryptographic key to a biometric, so that neither the key nor the biometric can be retrieved from the stored template. The key is re-created only if the correct live biometric sample is presented on verification.

The digital key (password, PIN, etc.) is randomly generated on enrolment, so that the user (or anybody else for that matter) does not even know about it. The key itself is completely independent of biometrics and, therefore, can always be changed or updated. After a biometric sample is acquired, the biometric encryption algorithm securely and consistently binds the key to the biometric to create a protected biometric encryption template, also called ‘private template’. In essence, the key is *encrypted* with the biometric. The biometric encryption template provides an excellent privacy protection and can be stored either in a database or locally (smart card, token, laptop, cell phone, etc.). At the end of the enrolment, both the key and the biometric are discarded.

For biometric decryption, the user presents a fresh biometric sample. This, when applied to the legitimate biometric encryption template, will let the biometric encryption algorithm retrieve the same key/password. In other words, the biometric serves as a *decryption key*. At the end of verification, the biometric sample is discarded once again. The biometric encryption algorithm is designed to account for acceptable variations in the input biometric. On the other hand, an attacker, whose biometric sample is different enough, will not be able to retrieve the password.

9.18 Cryptanalysis

LO 4

DID YOU KNOW ? Cryptanalysis is the science (or black art!) of recovering the plaintext of a message from the ciphertext without access to the key. In cryptanalysis, it is always assumed that the cryptanalyst has full access to the algorithm. An attempted cryptanalysis is known as an attack, of which there are the following types.

- *Brute force attack*: This technique requires a large amount of computing power and a large amount of time to run. It consists of trying all possibilities in a logical manner until the correct one is found. For

the majority of encryption algorithms, a brute force attack is impractical due to the large number of possibilities.

- *Ciphertext-only*: The only information the cryptanalyst has to work with is the ciphertext of various messages which are all encrypted with the same algorithm.
- *Known-plaintext*: In this scenario, the cryptanalyst has access not only to the ciphertext of various messages, but also the corresponding plaintext as well.
- *Chosen-plaintext*: The cryptanalyst has access to the same information as in a known-plaintext attack, but this time may choose the plaintext that gets encrypted. This attack is more powerful, as specific plaintext blocks can be chosen that may yield more information about the key. An adaptive-chosen-plaintext attack is merely one where the cryptanalyst may repeatedly encrypt plaintext, thereby modifying the input based on the results of a previous encryption.
- *Chosen-ciphertext*: The cryptanalyst uses a relatively new technique called differential cryptanalysis, which is an interactive and iterative process. It works through many rounds using the results from previous rounds, until the key is identified. The cryptanalyst repeatedly chooses ciphertext to be decrypted, and has access to the resulting plaintext. From this they try to deduce the key.
- *Pattern attack*: The cryptanalyst looks for patterns in the ciphertext to break the cipher. Therefore, good ciphers make the ciphertext appear as random as possible.
- *Statistical attack*: The cryptanalyst uses the inherent characteristics of the plaintext. For example, certain alphabets/combinations in the English language occur more frequently (E, S, T, QU, THE, ING, etc.)

There is only one totally secure algorithm, the one-time pad. All other algorithms can be broken given infinite time and resources. Modern cryptography relies on making it computationally unfeasible to break an algorithm. This means that while it is theoretically possible, the time-scale and resources involved make it completely unrealistic.

If an algorithm is presumed to be perfect, then the only method of breaking it relies on trying every possible key combination until the resulting ciphertext makes sense. As mentioned above, this type of attack is called a brute-force attack. The field of parallel computing is perfectly suited to the task of brute force attacks, as every processor can be given a number of possible keys to try, and they do not need to interact with each other at all except to announce the result. A technique that is becoming increasingly popular is parallel processing using thousands of individual computers connected to the Internet. This is known as distributed computing. Many cryptographers believe that brute force attacks are basically ineffective when long keys are used. An encryption algorithm with a large key (over 100 bits) can take millions of years to crack, even with powerful, networked computers of today. Besides, adding a single extra key doubles the cost of performing a brute force cryptanalysis.

Regarding brute force attack, there are a couple of other pertinent questions. What if the original plaintext is itself a cipher? In that case, how will the hacker know if he has found the right key. In addition, is the cryptanalyst sitting at the computer and watching the result of each key that is being tested? Thus, we can assume that brute force attack is impossible provided long enough keys are being used.

Here are some of the techniques that have been used by cryptanalysts to attack ciphertext.

- *Differential cryptanalysis*: As mentioned before, this technique uses an iterative process to evaluate cipher that has been generated using an iterative block algorithm (e.g., DES). Related plaintext is

encrypted using the same key. The difference is analysed. This technique proved successful against DES and some hash functions.

- *Linear cryptanalysis*: In this, pairs of plaintext and ciphertext are analysed and a linear approximation technique is used to determine the behaviour of the block cipher. This technique was also used successfully against DES.
- *Algebraic attack*: This technique exploits mathematical structure in block ciphers. If the structure exists, a single encryption with one key might produce the same result as a double encryption with two different keys. Thus the search time can be reduced.

However, strong or weak, the algorithm used to encrypt it, a message can be thought of as secure if the time and/or resources needed to recover the plaintext greatly exceed the benefits bestowed by having the contents. This could be because the cost involved is greater than the financial value of the message, or simply that by the time the plaintext is recovered the contents will be outdated.

9.19 Politics of Cryptography

LO 4

DID YOU KNOW ? Widespread use of cryptosystems is something most governments are not particularly happy about—precisely because it threatens to give more privacy to the individual, including criminals. For many years, police forces have been able to tap phone lines and intercept mail, but in an encrypted future that may become impossible.

This has led to some strange decisions on the part of governments, particularly the United States government. In the United States, cryptography is classed as munitions and the export of programs containing cryptosystems is tightly controlled. In 1992, the Software Publishers Association reached an agreement with the State Department to allow the export of software that contained RSA's RC2 and RC4 encryption algorithms, but only if the key size was limited to 40 bits as opposed to the 128 bit keys available for use within the US. This significantly reduced the level of privacy produced. In 1993, the US Congress had asked the National Research Council to study U.S. cryptographic policy. Its 1996 report, the result of two years' work, offered the following conclusions and recommendations.

- “On balance, the advantages of more widespread use of cryptography outweigh the disadvantages.”
- “No law should bar the manufacture, sale or use of any form of encryption within the United States.”
- “Export controls on cryptography should be progressively relaxed but not eliminated.”

In 1997, the limit on the key size was increased to 56 bits. The US government has proposed several methods whereby it would allow the export of stronger encryption, all based on a system where the US government could gain access to the keys if necessary, for example the *clipper* chip. Recently there have been lots of protests from the cryptographic community against the US government imposing restrictions on the development of cryptographic techniques. The article by Ronald L. Rivest, Professor, MIT, in the October 1998 issue of the *Scientific American*, (pages 116–117) titled “The Case against Regulating Encryption Technology,” is an example of such a protest. The resolution of this issue is regarded to be one of the most important for the future of **E-commerce**. Some of the key patents in the area of cryptography are listed in Table 9.3.



INDUSTRIAL RELEVANCE

Table 9.3 Some key patents in the area of cryptography

Inventors	Issue Year	Area
Feistel	1974	Lucifer cipher
Ehram et al.	1976	DES Algorithm
Hellman-Diffie-Merkle	1980	Key agreement
Merkle	1982	Tree authentication
Rivest-Shamir-Adleman	1983	RSA algorithm
Hellman-Bach	1986	Generation of strong primes
Brachtl et al.	1990	Hash functions
Massey-Lai	1993	IDEA block cipher
Kravitz	1993	Digital Signature Algorithm
Brickell et al.	1994	Exponentiation method

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 4:

- Suppose Alice and Bob use the Diffie-Hellman key exchange protocol with a common prime $P = 71$ and the primitive root $= 7$. If user A has private key $K_A = 5$, what is A's public key? If user B has private key $K_B = 12$, what is B's public key? What is the shared secret key?
- Show that the point $(4, 7)$ lies on the elliptic curve $y^2 = x^3 - 5x + 5$ over real numbers.
- Suppose the point $(a, 7)$ lies on the elliptic curve $y^2 = x^3 + 11x + 19 \pmod{167}$. Find the value of a .
- Show that the points $P = (3.5, 9.5)$ and $Q = (2.5, 8.5)$ lie on the elliptic curve over the real numbers: $y^2 = x^3 - 36x$. Find $P + Q$ and $2P$.
- What is the maximum period that can be obtained from the generator:
 $x_{n+1} = (ax_n) \pmod{247}$
For what value of a will it be achieved?

M

S

S

M

S

If you have successfully solved the above problems,
you have mastered LO 4. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/625>



9.20 Concluding Remarks

In this section we present a brief history of cryptography. People have tried to conceal information in written form since writing was developed. Examples survive in stone inscriptions and papyruses showing that many ancient civilisations including the Egyptians, Hebrews and Assyrians all developed cryptographic systems. The first recorded use of cryptography for correspondence was by the Spartans who (as early as 400 BC) employed a cipher device called a *scytale* to send secret communications between military commanders. The scytale consisted of a tapered baton around which was wrapped a piece of parchment inscribed with the message. Once unwrapped, the parchment appeared to contain an incomprehensible set of letters, however when wrapped around another baton of identical size the original text appeared.

The Greeks were, therefore, the inventors of the first transposition cipher and in the fourth century BC the earliest treatise on the subject was written by a Greek, Aeneas Tacticus, as part of a work entitled *On the Defence of Fortifications*. Another Greek, Polybius, later devised a means of encoding letters into pairs of symbols using a device known as the *Polybius checkerboard* which contains many elements common to later encryption systems. In addition to the Greeks there are similar examples of primitive substitution or transposition ciphers in use by other civilisations including the Romans. The Polybius checkerboard consists of a five-by-five grid containing all the letters of the alphabet. Each letter is converted into two numbers, the first is the row in which the letter can be found and the second is the column. Hence the letter A becomes 11, the letter B 12 and so forth.

The Arabs were the first people to clearly understand the principles of cryptography and to elucidate the beginning of cryptanalysis. They devised and used both substitution and transposition ciphers and discovered the use of letter frequency distributions in cryptanalysis. As a result of this, by approximately 1412, al-Kalkashandi could include in his encyclopaedia, *Subh al-a'sha*, a respectable, even if elementary, treatment of several cryptographic systems. He also gave explicit instructions on how to cryptanalyse ciphertext using letter frequency counts including examples illustrating the technique.

European cryptography dates from the Middle Ages during which it was developed by the Papal and Italian city states. The earliest ciphers involved only vowel substitution (leaving the consonants unchanged). Circa 1379 the first European manual on cryptography, consisting of a compilation of ciphers, was produced by Gabriele de Lavinde of Parma, who served Pope Clement VII. This manual contains a set of keys for correspondents and uses symbols for letters and nulls with several two-character code equivalents for words and names. The first brief code vocabularies, called nomenclatures, were expanded gradually and for several centuries were the mainstay of diplomatic communications for nearly all European governments. In 1470 Leon Battista Alberti described the first cipher disk in *Trattati in cifra*, and the *Traicté de chiffres*, published in 1586 by Blaise de Vigernère, contained a square table commonly attributed to him as well as descriptions of the first plaintext and ciphertext autokey systems.

By 1860 large codes were in common use for diplomatic communications and cipher systems had become a rarity for this application. However, cipher systems prevailed for military communications (except for high-command communications because of the difficulty of protecting codebooks from capture or compromise). During the US Civil War, the Federal Army extensively used transposition ciphers. The Confederate Army primarily used the Vigenère cipher and, occasionally, monoalphabetic substitution. While the Union cryptanalysts solved most of the intercepted Confederate ciphers, the Confederacy, in desperation, sometimes published Union ciphers in newspapers, appealing for help from readers in cryptanalysing them.

During the First World War both sides employed cipher systems almost exclusively for tactical communications while code systems were still used mainly for high-command and diplomatic

communications. Although field cipher systems such as the U.S. Signal Corps cipher disk lacked sophistication, some complicated cipher systems were used for high-level communications by the end of the war. The most famous of these was the German ADFGVX fractionation cipher.

In the 1920s the maturing of mechanical and electromechanical technology came together with the needs of telegraphy and radio to bring about a revolution in cryptodevices—the development of rotor cipher machines. The concept of the rotor had been anticipated in the older mechanical cipher disks. However, it was an American, Edward Hebern, who recognised that by hardwiring a mono-alphabetic substitution in the connections from the contacts on one side of an electrical rotor to those on the other side and cascading a collection of such rotors, polyalphabetic substitutions of almost any complexity could be produced. From 1921, and continuing through the next decade, Hebern constructed a series of steadily improving rotor machines that were evaluated by the U.S. Navy. It was undoubtedly this work which led to the United States' superior position in cryptology during the Second World War. At almost the same time as Hebern was inventing the rotor cipher machine in the United States, European engineers such as Hugo Koch (Netherlands) and Arthur Scherbius (Germany) independently discovered the rotor concept and designed the precursors to the most famous cipher machine in history, the German Enigma machine which was used during Second World War. These machines were also the stimulus for the TYPEX, the cipher machine employed by the British during Second World War.

The United States introduced the M-134-C (SIGABA) cipher machine during Second World War. The Japanese cipher machines of the Second World War have an interesting history linking them to both the Hebern and the Enigma machines. After Herbert Yardley, an American cryptographer who organised and directed the U.S. government's first formal code-breaking efforts during and after the First World War, published *The American Black Chamber* in which he outlined details of the American successes in cryptanalysing the Japanese ciphers, the Japanese government set out to develop the best cryptomachines possible. With this in mind, it purchased the rotor machines of Hebern and the commercial Enigmas, as well as several other contemporary machines, for study. In 1930 the first rotor machine of the Japanese, code named RED by U.S. cryptanalysts, was put into service by the Japanese Foreign Office. However, drawing on experience gained from cryptanalysing the ciphers produced by the Hebern rotor machines, the U.S. Army Signal Intelligence Service team of cryptanalysts succeeded in cryptanalysing the RED ciphers. In 1939 the Japanese introduced a new cipher machine, code-named PURPLE by U.S. cryptanalysts, in which the rotors were replaced by telephone-stepping switches. The greatest triumphs of cryptanalysis occurred during the Second World War when the Polish and British cracked the Enigma ciphers and the American cryptanalysts broke the Japanese RED, ORANGE and PURPLE ciphers. These developments played a major role in the Allies' conduct of the Second World War.

After the Second World War the electronics that had been developed in support of radar were adapted to cryptomachines. The first electrical cryptomachines were little more than rotor machines where the rotors had been replaced by electronic substitutions. The only advantage of these electronic rotor machines was their speed of operation and they inherited the inherent weaknesses of the mechanical rotor machines. The publication of the paper 'Communication Theory of Secrecy Systems' by C. E. Shannon in 1949 ushered in the era of *scientific* secret-key cryptography.

The advancement in computing power has meant an unprecedented freedom for cipher designers to use elaborate designs which would be far too prone to error if handled by pencil and paper, or far too expensive to implement in the form of an electromechanical cipher machine. The main thrust of development has been in the development of block ciphers, beginning with the *LUCIFER* project at IBM, a direct ancestor of the *DES* (Data Encryption Standard).

A big thrust in the development of cryptographic techniques was provided by the publication of the seminal paper ‘New Directions in Cryptography’ by W. Diffie and M. E. Hellman in 1976. They showed, for the first time, that secret communication was possible without any transfer of a secret key between sender and receiver, thus establishing the turbulent epoch of public-key cryptography.

There is a place for both symmetric and public-key algorithms in modern cryptography. Hybrid cryptosystems successfully combine aspects of both and seem to be secure and fast. While PGP and its complex protocols are designed with the Internet community in mind, it should be obvious that the encryption behind it is very strong and could be adapted to suit many applications. There may well still be instances when a simple algorithm is necessary, and with the security provided by algorithms like IDEA, there is absolutely no reason to think of these as significantly less secure. Elliptic curve cryptography (ECC) was discovered in 1985 by Victor Miller (IBM) and Neil Koblitz (University of Washington) as an alternative mechanism for implementing public-key cryptography. Unlike other popular algorithms such as RSA, ECC is based on discrete logarithms that are much more difficult to challenge at equivalent key lengths.

The roots of quantum cryptography lay in a research proposal by Stephen Weisner, called ‘Conjugate Coding’, from the early 1970s. It was eventually published in 1983, and by that time Bennett and Brassard, who were familiar with Weisner’s ideas, were ready to publish ideas of their own. They proposed ‘BB84’, the first quantum cryptography protocol, in 1984, but it was not until 1991 that the first experimental prototype based on this protocol was made operable (over a distance of 32 centimetres). More recent systems have been tested successfully on fibre optic cables over distances in the kilometres.

Chaos theory, a branch of the theory of nonlinear dynamical systems, has been well researched as a candidate for cryptography. Low-dimensional dynamical systems are capable of complex and unpredictable behaviour, which is useful for diffusion and confusion of information. Recent momentum in this domain has been provided by Baptista, Kocarev and Bose.

Identification systems, based on a wide variety of biometric patterns, have been of interest to the academia, the industry and the science-fiction based movie-makers since early 1970s. The research community today is working with all the different types of biometrics: (i) traditional biometrics (e.g. fingerprint, hand geometry, iris, and retina), (ii) more recent biometric patterns (e.g., voice, signature, palm print, and face) and (iii) innovative approaches (e.g., ear shape, DNA, keystroke/typing-rhythm, asymmetry of the face and body odour). Because of its variability, the biometric pattern or template itself cannot serve as a cryptographic key. The exploration in area of biometric-based encryption started accelerating in the 1990s and is currently a hot research topic.

An article posted on the Internet on the subject of picking locks stated: ‘The most effective door opening tool in any burglar’s toolkit remains the crowbar’. This also applies to cryptanalysis—direct action is often the most effective. It is all very well transmitting your messages with 128-bit IDEA encryption, but if all that is necessary to obtain that key is to walk up to one of the computers used for encryption with a pen-drive (USB stick), then the whole point of encryption is negated. In other words, an incredibly strong algorithm is not sufficient. For a system to be effective, there must be effective management protocols involved.

Finally, in the words of Sir Edgar Allen Poe, ‘Human ingenuity cannot concoct a cipher which human ingenuity cannot resolve’.

LEARNING OUTCOMES

- A cryptosystem is a collection of algorithms and associated procedures for hiding and revealing information. Cryptanalysis is the process of analysing a cryptosystem, either to verify its integrity or to break it for ulterior motives. An attacker is a person or system that performs cryptanalysis in order to break a cryptosystem. The process of attacking a cryptosystem is often called cracking. The job of the cryptanalyst is to find the weaknesses in the cryptosystem.
- According to the Kerckhoff's Principle, a cryptosystem should remain secure even if everything about the system, except the key, is known.
- A message being sent is known as plaintext. The message is coded using a cryptographic algorithm. This process is called encryption. An encrypted message is known as ciphertext, and is turned back into plaintext by the process of decryption.
- A key is a value that causes a cryptographic algorithm to run in a specific manner and produce a specific ciphertext as an output. The key size is usually measured in bits. The bigger the key size, the more secure will be the algorithm.
- Symmetric algorithms (or single key algorithms or secret key algorithms) have one key that is used both to encrypt and decrypt the message, hence their name. In order to decrypt the message for the recipient, they need to have an identical copy of the key. This presents one major problem, the distribution of the keys.
- Block ciphers usually operate on groups of bits called blocks. Each block is processed multiple times. In each round the key is applied in a unique manner. The more the number of iterations, the longer is the encryption process, but results in a more secure ciphertext.
- Stream ciphers operate on plaintext one bit at a time. Plaintext is streamed as raw bits through the encryption algorithm. While a block cipher will produce the same ciphertext from the same plaintext using the same key, a stream cipher will not. The ciphertext produced by a stream cipher will vary under the same conditions.
- The avalanche effect is a property of any encryption algorithm such that a small change in either the plaintext or the key produces a significant change in the ciphertext.
- To determine how much security one needs, the following questions must be answered.
 1. How much is the data to be protected worth?
 2. How long does it need to be secure?
 3. What are the resources available to the cryptanalyst/hacker?
- Two symmetric algorithms, both block ciphers, have been discussed in this chapter. These are the Data Encryption Standard (DES) and the International Data Encryption Algorithm (IDEA).
- Advanced Encryption Standard (AES) is based on the Rijndael cipher developed by two cryptographers, Joan Daemen and Vincent Rijmen. Rijndael is a family of ciphers with different key and block sizes. AES is a symmetric-key algorithm, where the same key is used for both encrypting and decrypting the data.
- RC4 generates a pseudorandom stream of bits. This key-stream can be used for encryption by XORing it with the plaintext using bit-wise operation. The decryption is performed the same way.

- Public-key algorithms are asymmetric, that is to say the key that is used to encrypt the message is different to the key used to decrypt the message. The encryption key, known as the public key is used to encrypt a message, but the message can only be decoded by the person that has the decryption key, known as the private key. Rivest, Shamir and Adleman (RSA) algorithm is a popular public-key encryption technique.
- The hybrid algorithm, Pretty Good Privacy (PGP), has four main modules: a symmetric cipher—IDEA for message encryption, a public-key algorithm—RSA to encrypt the IDEA key and hash values, a one-way hash function—MD5 for signing, and a random number generator.
- RSA relies on the fact that it is easy to multiply two large prime numbers together, but extremely hard (i.e., time consuming) to factor them back from the result. Factoring a number means finding its prime factors, which are the prime numbers that need to be multiplied together in order to produce that number.
- A one-way hash function is a mathematical function that takes a message string of any length (pre-string) and returns a smaller fixed-length string (hash value). These functions are designed in such a way that not only it is very difficult to deduce the message from its hashed version, but also given that all hashes are of a certain length, it is extremely hard to find two messages that hash to the same value.
- If $p > 3$ is a prime number and E is an elliptic curve mod p , and $\mathbf{P}, \mathbf{Q} \in E$, the Elliptic Curve Discrete Logarithm problem is to find a positive integer m such that $\mathbf{Q} = m\mathbf{P}$, if such an integer exists.
- The Elliptic Curve Cryptography (ECC) relies on the difficulty of finding \mathbf{P} given $m\mathbf{P}$. Unlike other popular algorithms such as RSA, ECC is based on discrete logarithms that are much more difficult to challenge at equivalent key lengths.
- The Diffie-Hellman key agreement protocol (also called exponential key agreement) allows two users to exchange a secret key over an insecure medium without any prior secrets.
- Chaos functions can be used for secure communications and cryptographic applications. The chaotic functions are primarily used for generating keys that are essentially unpredictable.
- In Quantum Cryptography, the information is protected by the laws of physics and provides means for two parties to exchange an enciphering key over a private channel with complete security of communication. The Heisenberg Uncertainty Principle and quantum entanglement is exploited in such a system.
- Biometric Encryption is a process that securely binds a PIN or a cryptographic key to a biometric, so that neither the key nor the biometric can be retrieved from the stored template. The key is re-created only if the correct live biometric sample is presented on verification.
- An attempted cryptanalysis is known as an attack, of which there are five major types: Brute force attack, Ciphertext-only, Known-plaintext, Chosen-plaintext and Chosen-ciphertext.
- The common techniques that are used by cryptanalysts to attack ciphertext are differential cryptanalysis, linear cryptanalysis and algebraic attack.
- Widespread use of cryptosystems is something most governments are not particularly happy about, because it threatens to give more privacy to the individual, including criminals.

Imagination is more important than knowledge

Albert Einstein (1879–1955)



MULTIPLE CHOICE QUESTIONS

- 9.1 A cryptosystem is
 - (a) A set of protocols for analysing information
 - (b) A collection of algorithms and associated procedures for hiding and revealing information
 - (c) A set of procedures for distorting information
 - (d) None of the above
- 9.2 Cryptanalysis is
 - (a) The process of analysing a cryptosystem
 - (b) The process of hacking a cryptosystem
 - (c) The process of synthesizing a cryptosystem
 - (d) None of the above
- 9.3 According to the Kerckhoff's Principle, a cryptosystem should remain secure even if
 - (a) The key of the cryptosystem is known to the hacker
 - (b) The hacker has infinite resources
 - (c) Everything about the cryptosystem, except the key, is known
 - (d) None of the above
- 9.4 Following is true about a key
 - (a) The key size is usually measured in bits
 - (b) The bigger the key size, the more secure will be the algorithm
 - (c) Both (a) and (b)
 - (d) None of the above
- 9.5 Following is true about symmetric key algorithms
 - (a) The same key is used both to encrypt and decrypt the message
 - (b) They are also known as secret key algorithms
 - (c) They are also known as single key algorithms
 - (d) All of the above
- 9.6 Choose the correct statement
 - (a) Block ciphers usually operate on groups of bits called blocks
 - (b) Stream ciphers operate on plaintext one bit at a time
 - (c) A block cipher produces the same ciphertext from the same plaintext using the same key while a stream cipher may not produce the same ciphertext
 - (d) All of the above
- 9.7 The avalanche effect is a property of any encryption algorithm such that
 - (a) A small change in the plaintext produces a significant change in the ciphertext
 - (b) A small change in the key produces a significant change in the ciphertext

- (c) Both (a) and (b)
 - (d) None of the above
- 9.8 The security of RSA relies on the fact that
- (a) It is easy to multiply two large prime numbers, but difficult to factor them back
 - (b) It is easy to factorise a number into its prime factors
 - (c) It is easy to add several large prime numbers
 - (d) None of the above
- 9.9 The security of Elliptic Curve Cryptography relies on
- (a) The difficulty of finding P given P^m
 - (b) The difficulty of finding P given mP
 - (c) The difficulty of finding P given $\log_m(P)$
 - (d) None of the above
- 9.10 The Diffie-Hellman key agreement protocol
- (a) Allows two users to exchange a secret key over an insecure medium without any prior secrets
 - (b) Allows two users to exchange a public key over an insecure medium without any prior secrets
 - (c) Allows two users to exchange a hash value over an insecure medium without any prior secrets
 - (d) None of the above

For interactive quiz with answers, scan the QR code given here

Or

Visit <http://qrcode.flipick.com/index.php/574>



SHORT-ANSWER TYPE QUESTIONS



- 9.1 What is the difference between a block cipher and a stream cipher?
- 9.2 What are the two general approaches to attacking a cipher?
- 9.3 What are the problems with the one-time pad?
- 9.4 What is the difference between diffusion and confusion?
- 9.5 What is the purpose of the S-boxes in DES?
- 9.6 What is the avalanche effect?
- 9.7 What is the difference between differential and linear cryptanalysis?
- 9.8 What are the important design considerations for a stream cipher?
- 9.9 Why is it not desirable to reuse a stream cipher key?
- 9.10 What primitive operations are used in RC4?
- 9.11 What is a nonce?
- 9.12 What are the roles of the public and private keys?
- 9.13 What is a one-way function?
- 9.14 What is an elliptic curve and what is the zero point of the elliptic curve?
- 9.15 Give two reasons why a message authentication code should be derived from a cryptographic hash function as opposed to a symmetric cipher?

For answers, scan the QR code given here

Or

Visit <http://qrcode.flipick.com/index.php/626>



PROBLEMS

- S** 9.1 We want to test the security of *character + x* encrypting technique in which each alphabet of the plaintext is shifted by *x* to produce the ciphertext.
- How many different attempts must be made to crack this code assuming brute force attack is being used?
 - Assuming it takes a computer 1 ms to check out one value of the shift, how soon can this code be broken into?
- M** 9.2 Decrypt the following encrypted sequence which uses the *character + x* encrypting technique (only the 26 characters from the English alphabet have been used). Note: This problem can be solved without using a computer!

JXU SQJ YI EKJ EV JXU RQW

- S** 9.3 Transposition ciphers rearrange the letters of the plaintext without changing the letters themselves. For example, a very simple transposition cipher is the *rail fence*, in which the plaintext is staggered between two rows and then read off to give the ciphertext. In a two row rail fence the message MERCHANT TAYLORS' SCHOOL becomes:

M	R	H	N	T	Y	O	S	C	O	L
E	C	A	T	A	L	R	S	H	O	

Which is read out as: MRHNTYOSCOLECATALRSHO.

- If a cryptanalyst wants to break into the rail fence cipher, how many distinct attacks must he make, given the length of the ciphertext is *n*?
 - Suggest a decrypting algorithm for the rail fence cipher.
- M** 9.4 One of the most famous field ciphers ever was a fractionation system—the *ADFGVX cipher*, which was employed by the German Army during the First World War. This system was so named because it used a 6×6 matrix to substitution-encrypt the 26 letters of the alphabet and 10 digits into pairs of the symbols A, D, F, G, V and X. The resulting bilateral cipher is only an intermediate cipher, it is then written into a rectangular matrix and transposed to produce the final cipher which is the one which would be transmitted. Here is an example of enciphering the phrase “Merchant Taylors” with this cipher using the key word “Subject”.

	A	D	F	G	V	X
A	S	U	B	J	E	C
D	T	A	D	F	G	H
F	I	K	L	M	N	O
G	P	Q	R	V	W	X
V	Y	Z	0	1	2	3
X	4	5	6	7	8	9

Plaintext: M E R C H A N T
Ciphertext: FG AV GF AX DX DD FV DA

T A Y L O R S
DA DD VA FF FX GF AA

This intermediate ciphertext can then be put in a transposition matrix based on a different key.

C	I	P	H	E	R
1	4	5	3	2	6
F	G	A	V	G	F
A	X	D	X	D	D
F	V	D	A	D	A
D	D	V	A	F	F
F	X	G	F	A	A

The final cipher is therefore: FAFDFGDDFAVXAAFGXVDXADDVGFDAFA.

- (i) If a cryptanalyst wants to break into ADFGVX cipher, how many distinct attacks must he make, given the length of the ciphertext is n ?
- (ii) Suggest a decrypting algorithm for the ADFGVX cipher.

- M** 9.5 Consider the knapsack technique for encryption proposed by Ralph Merkle of XEROX and Martin Hellman of Stanford University in 1976. They suggested using the knapsack, or subset-sum, problem as the basis for a public key cryptosystem. This problem entails determining whether a number can be expressed as a sum of some subset of a given sequence of numbers and, more importantly, which subset has the desired sum.

Given a sequence of numbers A , where $A = (a_1 \dots a_n)$, and a number C , the knapsack problem is to find a subset of $a_1 \dots a_n$ which sums to C .

Consider the following example:

$$n = 5, C = 14, A = (1, 10, 5, 22, 3)$$

$$\text{Solution} = 14 = 1 + 10 + 3$$

In general, all the possible sums of all subsets can be expressed by:

$$m_1 a_1 + m_2 a_2 + m_3 a_3 + \dots + m_n a_n \text{ where each } m_i \text{ is either 0 or 1.}$$

The solution is therefore a *binary vector* $M = (1, 1, 0, 0, 1)$.

There is a total number 2^n of such vectors (in this example $2^5 = 32$).

Obviously not all values of C can be formed from the sum of a subset and some can be formed in more than one way. For example, when $A = (14, 28, 56, 82, 90, 132, 197, 284, 341, 455, 515)$ the number 515 can be formed in three different ways but the number 516 cannot be formed in any ways.

- (i) If a cryptanalyst wants to break into this knapsack cipher, how many distinct attacks must he make?
 - (ii) Suggest a decrypting algorithm for the knapsack cipher.
- S** 9.6 Encode the following plaintext using Vigenère cipher: CRYPTOGRAPHY IS FUN. Use the key QUANTUM.
- S** 9.7 Decode the following Vigenère ciphertext: QQNLMEPQBVLBI. Use the key: IIT.

- M** 9.8 Bob is using an affine cipher to send Alice a ciphertext TURGIFSFSFERBI using the mod 26 alphabet. Alice has lost the key of this cipher but she guesses that the first two letters are CR. Can Alice decrypt the ciphertext?
- M** 9.9 Suppose we have N songs stored in our music player and we play the songs in ‘shuffle’ mode, so that a song is randomly chosen and played. Approximately how many songs will be played before we will hear some song twice (with probability about 1/2)? How is this problem related to cryptography?
- S** 9.10 Consider a 3-D cipher, which works as follows. A cube is constructed with 3 blocks on each edge so that there are a total of 27 blocks inside the cube. Each block represents one letter of the alphabet or a space (so, 27 characters to be assigned to the 27 blocks). Thus, every letter has three coordinate points (α, β, γ) which take values from the set $\{1, 2, 3\}$. One possible cipher may be: ‘a’ = $(1, 1, 1)$, ‘b’ = $(1, 1, 2)$, ..., ‘z’ = $(3, 3, 2)$, ‘<space>’ = $(3, 3, 3)$. For an attacker using brute force technique, how many attempts will it take for an unknown 3-D cipher?
- M** 9.11 (i) Use the prime numbers 29 and 61 to generate keys using the RSA algorithm.
(ii) Represent the letters ‘RSA’ in ASCII and encode them using the key generated above.
(iii) Next generate keys using the pair of primes, 37 and 67. Which is more secure, the keys in part (i) or part (iii)? Why?
- S** 9.12 How many prime numbers are there between 10^{90} and 10^{100} ?
- S** 9.13 Consider a fully connected network consisting of n nodes. How many keys have to be generated for the entire network if all nodes are required to communicate with each other using
(i) Secret key encryption?
(ii) Public key encryption?
- M** 9.14 Consider the elliptic curve $y^2 \equiv x^3 + x + 1$ defined over \mathbb{Z}_{23} .
(i) We wish to carry out repeated self-addition using this curve. For $P = (3, 10)$, find the value of $2^{(10^{74}+1)}$.
(ii) Find the probability: Prob $\left[\lim_{m \rightarrow \infty} \{2^m P\} = (5, 19) \right]$.
- S** 9.15 We wish to determine the security of the jigsaw transform for image encryption. Suppose, the original image (size $p \times p$) is divided into sub-blocks (each $q \times q$) such that $p = mq$. These sub-blocks are randomly permuted and the ‘jig sawed’ image is obtained.
(i) Calculate the mathematical complexity of a brute force attack on this jigsaw encoding.
(ii) What is the key size (in bits) in this case?
- M** 9.16 Compute $7^{64} \bmod 23$ and $7^{71} \bmod 23$.
- S** 9.17 Apply Fermat’s primality test to see if $n = 721$ is a prime number.
- S** 9.18 Suppose, for a RSA-based cryptosystem, $A = 37$ and $B = 41$. Can the public key, $E = 49$ be a good choice?
- M** 9.19 Alice and Bob agree to use the prime $p = 1373$ and the base $g = 2$ for a Diffie-Hellman key exchange. Alice sends Bob the base $A = 974$. Suppose, Bob uses the secret exponent $b = 871$. What value B should Bob send to Alice? What is their secret shared key?
- S** 9.20 Consider the elliptic curve given by E : $y^2 \equiv x^3 + 22x + 153 \pmod{307}$. Check whether (i) $P = (167, 118) \in E$ and (ii) $Q = (220, 287) \in E$.
- D** 9.21 Consider the elliptic curve given by E : $y^2 \equiv x^3 + 17$ over the real number field. Check whether $P = (-1, 4)$ and $Q = (2, 5) \in E$. Find $P + Q$, $P - Q$ and $2P$.

- D** 9.22 Find all the points on the elliptic curve E : $y^2 \equiv x^3 + 2x + 7 \pmod{11}$. Make the addition table for all points.
- S** 9.23 Data compression is often used in data storage. Suppose we want to combine data compression with encryption. We have two options:
- Compress the data and then encrypt the result, or
 - Encrypt the data and then compress the result.
- Which one is better and why?

COMPUTER PROBLEMS



- Write a program that performs encryption using the DES.
- Write a program to encode and decode using IDEA. Compare the number of computations required to encrypt a plaintext using the same key-size for DES and IDEA.
- Write a general program that can factorise a given number. Plot a graph of the average number of floating point operations required versus the number of digits in the input that is being factorised.
- Write a program to encode and decode using the RSA algorithm. Plot the number of floating point operations required to be performed by the program versus the key-size.
- Write a program that implements the Diffie-Hellman protocol.
- Consider the difference equation

$$x_{n+1} = ax_n(1 - x_n)$$

For $a = 4$, this function behaves like a chaos function.

- Plot a sequence of 100 values obtained by iterative application of the difference equation. What happens if the starting values $x_0 = 0.5$?
- Take two initial conditions (i.e., two different starting values, x_{01} and x_{02}) which are separated by Δx . Use the difference equation to iterate each starting point n times and obtain the final values y_{01} and y_{02} , which are separated by Δy . For a given Δx , plot Δx versus n .
- For a given value of n (say $n = 500$), plot Δx versus Δy .
- Repeat parts (i), (ii) and (iii) for $a = 3.7$ and $a = 3.9$. Compare and comment.

PROJECT IDEAS

- Cryptanalysis of Simple Substitution Cipher:** Carry out the frequency analysis of the alphabet in the English language. Based on the relative frequency of the occurrences of different alphabets design a method to break a simple substitution cipher. Improve your design by looking at the frequency of two-letter and three-letter combinations.
- Cryptanalysis of Playfair Cipher:** Develop a software tool that can carry out the cryptanalysis of Playfair Cipher using the frequency analysis of digraphs.
- DES and AES:** We wish to implement the DES and AES in hardware. Give the circuit realisation of DES and AES in terms of adders, multipliers, logic-gates and memory units. Make assumption regarding the power consumed by the different circuit elements and comment on the relative power efficiencies of DES and AES.

- 9.4 **Triple DES:** In order to beat the ‘meet-in-the middle’ attack, we wish to execute the DES encryption three times with three different keys. For this triple DES, a plaintext block is first encrypted by passing it through an encryption algorithm. The result is then passed through the same encryption algorithm again, with a different key. The result of the second encryption is passed through the same encryption algorithm a third time, again with a different key. Implement triple DES in software and check its robustness against ‘meet-in-the middle’ attack.
- 9.5 **Chaos for Encryption:** One of the challenges in cryptography is generating seemingly random number sequences for use as encryption keys. Use a bank of chaos functions to generate random number sequences. Design a good algorithm to choose which chaos function to use.
- 9.6 **A Company with Two Directors:** Design a cryptographic scheme that requires two company directors to sign a contract. The contract is sent to both directors. They each sign and send their signatures back to the company secretary. The company secretary then assembles the two signatures into a valid signature on the contract. Note that the two directors communicate with the secretary, but are not allowed to communicate with each other.

REFERENCES FOR FURTHER READING

1. Stallings, W., *Cryptography and Network Security: Principles and Practice*, (4th Edition), Prentice Hall, 2006.
2. Schneier, B., *Applied Cryptography: Protocols, Algorithms and Source Code in C*, John Wiley & Sons (2nd Edition), 1995.

Physical Layer Security

Distrust and caution are the parents of security.

Benjamin Franklin (1706–1790)

Learning objectives

After studying this chapter the students will be able to:

- LO 1** Explain the basic idea behind Shannon's security, wiretap model, degraded wiretap model and calculate secrecy capacity.
- LO 2** Describe the Gaussian wiretap model, calculate the secrecy capacity of wireless channels, define outage capacity and the outage probability.
- LO 3** Explain the idea behind cooperative jamming, artificial noise forwarding and solve problems related to friendly jammer with single and multiple antennas.

10.1 Introduction to Physical Layer Security

The approach presented in Chapter 9 consisted of cryptographic algorithms to ensure that only legitimate users can correctly interpret messages. This computational complexity-based cryptography assumes that the eavesdropper has finite computing and time resources. In this approach, both the legitimate receiver and the eavesdropper receive *exactly* the same encrypted message. The eavesdropper also knows the encoding and decoding process. The secret key is available only to the transmitter and the legitimate receiver. The difficulty in decrypting the received signal is based on a known difficult problem, coupled with the eavesdropper's finite computing and time resources. The basic problem with this approach is that if the adversary gains access to more sophisticated computing resources, the security of the cryptosystem is likely to be compromised. In this era of exascale computing, coupled with the possibility of massive networking, the threat is genuine.

...
This chapter comes with a video overview by the author. Scan here to know more or
Visit <http://qrcode.flipick.com/index.php/633>



There exists another school of thought which deals with information-theoretic security, where the eavesdropper is assumed to have unlimited computational and time resources. This information-theoretic encryption was first proposed by Shannon in 1949. The objective is to ensure that no information is released to the eavesdropper. In many practical systems (for example, wireless communication systems), the eavesdropper may have a worse channel (noisier channel) than the receiver. The secrecy of such systems can be analysed under the above assumption.

In this chapter

We will start with Shannon's notion of security and then define the wiretap model and the degraded wiretap model. The concept of strong secrecy, weak secrecy and equivocation rate will also be introduced. The idea of secrecy capacity will be explained, in LO 1.

Next, we shall discuss the Gaussian wiretap model and derive the secrecy capacity of the Gaussian wiretap channel. We will then study more practical wireless scenarios. Noisier and more capable channels will be discussed. We will also define outage capacity and the outage probability, in LO 2.

Finally, we will study cooperative jamming techniques to improve physical layer security. We will consider friendly jammers with single and multiple antennas. Artificial noise injection, as a means for increasing security, will also be discussed, in LO 3.

10.2 Shannon's Notion of Security

LO 1



Explain the basic idea behind Shannon's security, wiretap model, degraded wiretap model and calculate secrecy capacity.

The foundation for the theory of secrecy systems was laid by Claude Shannon in 1949. His model for a cipher system is depicted in Fig. 10.1. He assumed that both the main and eavesdropper's channels to be noiseless. As was common in those days, he also assumed the availability of a secret key, K , at both the transmitter and legitimate receiver. Shannon called the communication over the main channel *perfectly secure* if, and only if the entropy of the message M , given the eavesdropper's observation Z , to be equal to the entropy of the original message M , i.e.,

$$H(M|Z) = H(M) \quad (10.1)$$

This simply implies that Z does not contain *any* information about M . Since the mutual information can be expressed as

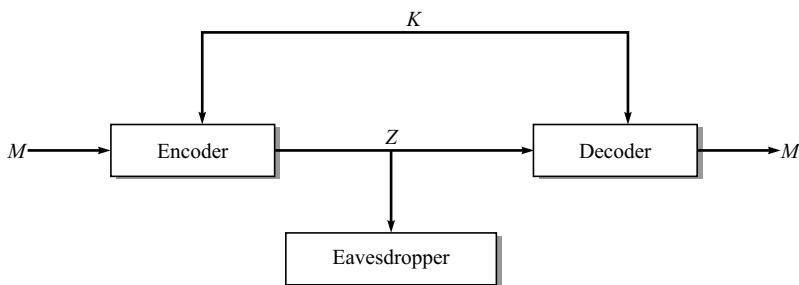


Fig. 10.1 Shannon's cipher model.

$$I(M; Z) = H(M) - H(M|Z) \quad (10.2)$$

we conclude that $I(M; Z) = 0$ for perfect secrecy. In other words, perfect secrecy is achieved if codewords Z are statistically independent of messages, M . Here, no assumptions are made regarding

- The detection strategy employed by the eavesdropper, and
- The computational power available to the eavesdropper.

Definition 10.1 The **Leakage of Information** to the eavesdropper is given by $I(M; Z)$.

Shannon then proved that perfectly secure communication can take place if, and only if the entropy of the shared secret key K is at least equal to that of the message, i.e.,

$$H(K) \geq H(M) \quad (10.3)$$

This implies that it is necessary to use at least one secret-key bit for each message bit to achieve perfect secrecy. For the case when both the key and message are binary encoded, perfect secrecy can be achieved when $H(K) \geq H(M)$ through a strategy that involves **One-Time-Pads**, proposed by Vernam in 1926 (see Fig. 10.2). This formed the basis for the secret-key cryptography, also called the private key cryptography. This also had the additional problem of key distribution. The number of keys also increased with the number of nodes in the network. The alternate avatar, the public-key cryptography, was based on computational advantage of the legitimate user. Public-key cryptography used the fact that certain mathematical operations are ‘easier in one direction and more difficult in the other’. For example, multiplication is easy while factorisation is difficult (RSA) or exponentiation is easy while discrete logarithm is difficult (Diffie-Hellman).

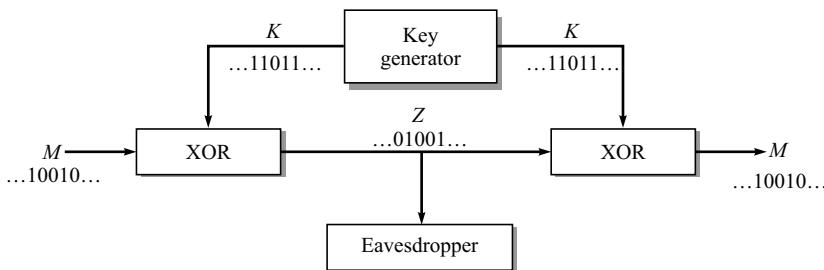
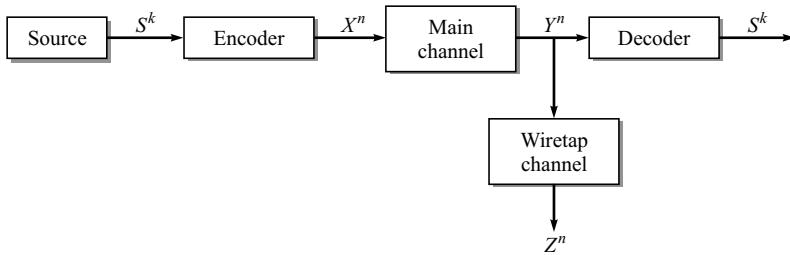


Fig. 10.2 Vernam’s cipher model.

10.3 The Wiretap Model

LO 1

The initial work by Shannon was revisited almost 25 years later by Wyner. The **wiretap model**, proposed by Wyner, is shown in Fig. 10.3. Here, X , is the transmitted signal and Y is the received signal at the legitimate receiver. In this model, the eavesdropper puts a ‘wire-tap’ on the receiver’s channel, and hence, receives a *degraded* version, Z , of the signal. The key property that enables secret communication in this case, even in the absence of a shared secret key, is that the eavesdropper’s channel is noisier than the receiver’s channel. The channel between the transmitter and the eavesdropper is called the *eavesdropper’s channel*.

**Fig. 10.3** Wyner's wiretap model.

Wyner provided two relaxations in the earlier assumptions made by Shannon which are as follows.

- **The noiseless communication assumption:** Wyner considered a noisy main channel and a noisy eavesdropper channel. This is the *noisy channel assumption* as opposed to the noiseless communication assumption made by Shannon.
- **The perfect secrecy assumption:** Wyner required the leakage of information at the eavesdropper to go to zero, when normalised by the blocklength, i.e., $\lim_{n \rightarrow \infty} \frac{1}{n} I(M; Z^n) \rightarrow 0$, where M is the message and Z^n is the observation by the eavesdropper, which is a vector of length n . This is the *weak secrecy constraint*, as opposed to the *perfect secrecy* assumption made by Shannon, depicted by (10.1).

Definition 10.2 Strong Secrecy implies that the message and the eavesdropper's observations are almost independent while **Weak Secrecy** implies that the normalised mutual information between the message and the eavesdropper vanishes, i.e.,

$$\begin{aligned} \lim_{n \rightarrow \infty} I(M; Z^n) &\rightarrow 0 && (\text{Strong Secrecy}) \\ \lim_{n \rightarrow \infty} \frac{1}{n} I(M; Z^n) &\rightarrow 0 && (\text{Weak Secrecy}) \end{aligned} \quad (10.4)$$

Note that, in this model, the presence of the ‘wiretapper’ is known to the transmitter and the legitimate user *a-priori*. The aim is to carry out secure communications in the presence of the eavesdropper. In classical ‘wiretapping’, once the wiretap was discovered, the communication was stopped until the wiretap was disengaged. Let us now look at this model in more detail.

Let the secret message, M , be a random integer taken from the set $\{1, 2, \dots, 2^{nR}\}$. This message is transmitted using the channel n times (i.e., n channel uses). If all the elements of the set are equi-probable, then the entropy of the message is given by

$$H(M) = nR \quad (10.5)$$

and the (secrecy) communication rate can be written as

$$R = \frac{H(S)}{n} \text{ bits per channel use.} \quad (10.6)$$

In these n instances of channel use, the transmitter sends the coded signal

$$X^n = X_1, X_2, \dots, X_n \quad (10.7)$$

and the legitimate receiver receives

$$Y^n = Y_1, Y_2, \dots, Y_n \quad (10.8)$$

Since the main channel is noisy, the receiver decodes the received message with error probability

$$P_e = \Pr[\hat{M} \neq M] \quad (10.9)$$

where \hat{M} is the decoded message.

The eavesdropper's channel, which is the composite of the main channel and the wiretap channel, is simply a degraded version of the main channel. This type of channel is also called the *degraded wiretap channel* (DWTC). The eavesdropper's channel can be represented as

$$p(Z^n | X^n) = p(Z^n | Y^n)p(Y^n | X^n) \quad (10.10)$$

Let the message overheard by the eavesdropper be

$$Z^n = Z_1, Z_2, \dots, Z_n \quad (10.11)$$

Then the residual uncertainty regarding the secret message M , having received Z^n , is given by the conditional entropy $H(M|Z^n)$.

Definition 10.3 A $(2^{nR}, n)$ code for a **Degraded Wiretap Channel** (DWTC) consists of

- (i) a random source at the encoder
- (ii) a message set M of size 2^{nR}
- (iii) an encoding function, f , which maps a message $m \in M$ to the codeword X^n , i.e., $m \rightarrow X^n$.
- (iv) a decoding function, g , which maps the channel observation Z^n to a message, i.e., $Z^n \rightarrow \hat{m}$.

Definition 10.4 The **Equivocation**, which is a measure of the confusion created at the eavesdropper, is given by $H(M|Z^n)$. If a code, C_n , with blocklength n is used, we can write the equivocation as

$$E(C_n) = H(M|Z^n C_n) \quad (10.12)$$

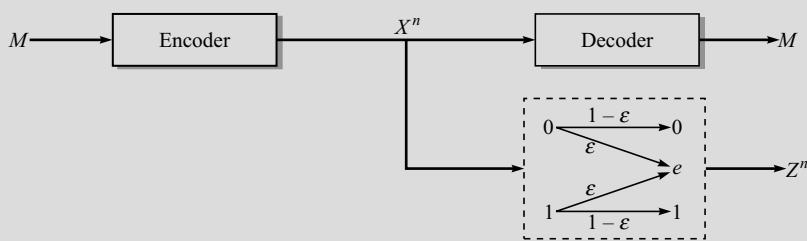
We note that the equivocation is explicitly conditioned on C_n , because it is assumed that the eavesdropper has the knowledge about the code, but does not know which specific codeword was transmitted. In this framework, the **Leakage of Information** to the eavesdropper is expressed as

$$L(C_n) = I(M; Z^n | C_n) \quad (10.13)$$

However, we must also ensure that the message is communicated reliably, where the reliability is measured in terms of the average probability of error

$$P_e(C_n) = \Pr[\hat{M} \neq M | C_n] \quad (10.14)$$

Example 10.1 Consider a binary erasure wiretap channel (with erasure probability ε), as shown in Fig. 10.4.

**Fig. 10.4** A binary erasure wiretap channel.

The message, M , is encoded as a binary codeword, X^n , of length n with $X \in \{0, 1\}$. The eavesdropper receives a vector Z^n of length n , with $Z \in \{0, 1, e\}$. It is assumed that different messages are always encoded into different codewords, so that the transmission rate is $(1/n)H(M) = (1/n)\log M$. Suppose, there are only two possible messages to be communicated, i.e., messages belonging to the set $\{1, 2\}$. Then, $H(M) = 1$. Let the encoding strategy be as follows: message 1 is sent using a binary sequence of length n with odd parity and message 2 is sent using binary sequences of length n with even parity. Observe that there can be several sequences of length n with either even or odd parity.

The eavesdropper will know the codeword perfectly, if no erasures occur. Otherwise, there will be confusion at the eavesdropper's end, since the parity of the received vector will be altered. We model this as a random variable

**Intuition**

Now, we consider the equivocation at the eavesdropper's end

$$H(M | Z^n) \geq H(M | Z^n, E) \quad (10.16)$$

because conditioning does not increase entropy. The probability that there are no erasures in the vector Z^n is

$$p_{NE} = (1 - \varepsilon)^n \quad (10.17)$$

while, the probability of at least one of the bits in erasure is $p_E = 1 - p_{NE} = 1 - (1 - \varepsilon)^n$.

We can write (10.16) as

$$\begin{aligned} H(M | Z^n, E) &= p_E H(M | Z^n, E = 1) + p_{NE} H(M | Z^n, E = 0) \\ &= (1 - (1 - \varepsilon)^n) H(M | Z^n, E = 1) + (1 - \varepsilon)^n H(M | Z^n, E = 0) \end{aligned} \quad (10.18)$$

Next, observe the following.

- (i) $H(M | Z^n, E = 1) = H(M)$ because even if one bit is in erasure there will be confusion at the eavesdropper's end and the entropy of the message will not be reduced even after receiving Z^n .
- (ii) $H(M | Z^n, E = 0) = 0$, because if there are no erasures, the eavesdropper will have complete knowledge of the message and the uncertainty (entropy) of the message will be zero.

Using these observations, we can rewrite (10.18) as

$$H(M | Z^n, E) = (1 - (1 - \varepsilon)^n) H(M). \quad (10.19)$$

Using (10.16) and the fact $H(M) = 1$, we can write

$$I(M; Z^n) = H(M) - H(M | Z^n) \leq (1 - \varepsilon)^n \quad (10.20)$$

which vanishes exponentially as $n \rightarrow \infty$. Thus, the coding strategy is secure.

This example is illustrative because it shows that

- (i) *The encoding and decoding strategy forms a part of the security system and ultimately affects the amount of information leaked to the eavesdropper. Here we have employed a parity-based strategy and*
- (ii) *The equivocation is a direct result of the degraded wiretap channel and depends on how ‘degraded’ the wiretap channel is. Figure 10.5 shows that the amount of information leaked to the eavesdropper reduces as the erasure probability increases.*

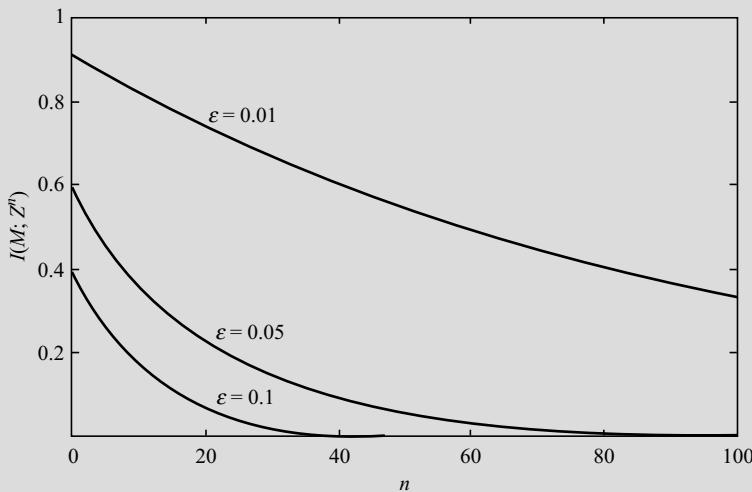


Fig. 10.5 Leakage of information versus the blocklength, n , for different erasure probabilities, ε .

Definition 10.5 The **Equivocation Rate** is the normalised equivocation, expressed as

$$\Delta = \frac{H(M | Z^n)}{H(M)}. \quad (10.21)$$

The higher the equivocation rate, the more secure is the communication. Consider the two extreme scenarios. Suppose, $H(M | Z^n) = H(M)$, i.e., the output of the eavesdropper’s channel (Z^n) conveys no information about the secret message, and hence does not reduce the uncertainty about M . In this case, $\Delta = 1$. On the other hand, if the output of the eavesdropper’s channel conveys everything about the secret message, i.e., $H(M | Z^n) = 0$, then $\Delta = 0$.

Thus, we have two objectives to fulfill simultaneously which are as follows.

- Reliable communication at a reasonable transmission rate, R
- Secret communication at a reasonable equivocation rate, Δ .

Definition 10.6 A Weak Rate-Equivocation Pair (R, R_e) is achievable for DWTC if there exists a sequence of $(2^{nR}, n)$ codes $\{C_n\}$ such that,

$$\begin{aligned} \lim_{n \rightarrow \infty} P_e(C_n) &= 0 \\ \lim_{n \rightarrow \infty} \frac{1}{n} E(C_n) &\geq R_e \end{aligned} \quad (10.22)$$

The first condition is the *reliability condition* whereas the second condition pertains to the *weak secrecy condition*. Both these conditions may not get satisfied simultaneously. In order to increase reliability we add redundancy in a known manner (channel coding). However, adding too much redundancy may reduce secrecy (guess why?). Thus a balance is required, and this can be done by an appropriate choice of a coding scheme.

Definition 10.7 Weak Rate-Equivocation Region for DWTC is given by

$$\mathcal{R} = \text{cl}(\{(R, R_e) : (R, R_e) \text{ is achievable}\}), \quad (10.23)$$

where $\text{cl}(X)$ refers to the closure of the set X .

We make the following observations.

- If Rate-Equivocation Pair (R, R_e) is achievable, then the pair (R, R'_e) is also achievable if $R_e \geq R'_e$.
- The Rate-Equivocation Pair $(R, 0)$ is always achievable.

Definition 10.8 Consider a DWTC characterised by $X, Y, Z, p_{Z|Y}$ and $p_{Y|X}$. For any distribution p_X on X , we define the set $\mathcal{R}^{\text{DWTC}}(p_X)$ as

$$\mathcal{R}^{\text{DWTC}}(p_X) = \left\{ (R, R_e) : \begin{array}{l} 0 \leq R_e \leq R \leq I(X; Y) \\ 0 \leq R_e \leq I(X; Y|Z) \end{array} \right\} \quad (10.24)$$

where the joint distribution of X, Y, Z factorises as $p_X p_{Y|X} p_{Z|Y}$. Then the **Rate-Equivocation Region** of the DWTC is the convex region given by

$$\mathcal{R}^{\text{DWTC}} = \bigcup_{p_X} \mathcal{R}^{\text{DWTC}}(p_X). \quad (10.25)$$

A typical rate-equivocation region $\mathcal{R}^{\text{DWTC}}$ is shown in Fig. 10.6.

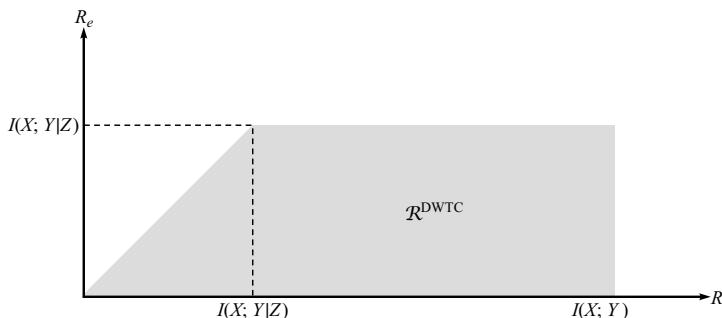


Fig. 10.6 A typical rate-equivocation region $\mathcal{R}^{\text{DWTC}}(p_X)$.

Definition 10.9 If Rate-Equivocation Pair (R, R_e) is achievable with $R = R_e$ then R is **Full Secrecy Rate**, i.e.,

$$R_e = \lim_{n \rightarrow \infty} \frac{1}{n} H(M | Z^n) = \lim_{n \rightarrow \infty} \frac{1}{n} H(M) = R \quad (10.26)$$

Full secrecy implies that the entire message is hidden from the eavesdropper and is sometimes referred to as **Perfect Secrecy**. However, observe that Shannon's definition of perfect secrecy, given by (10.1), is stricter, and requires exact statistical independence.



Definition 10.10 The **Secrecy Capacity** is the maximum possible transmission rate for which the eavesdropper is unable to decode any information. Here, the eavesdropper is assumed to have unlimited computing resources and time, and hence, the secrecy is *provable*.

In other words, secrecy capacity is the maximum rate at which secret information may be sent to the receiver, under perfect secrecy.

Definition 10.11 The **Strong Secrecy Capacity** of a DWTC is given by

$$C_s^{\text{DWTC}} = \sup_R \{R: (R, R) \in \mathcal{R}^{\text{DWTC}}\}. \quad (10.27)$$

Definition 10.12 The **Secrecy Capacity of a DWTC** is given by

$$C_s^{\text{DWTC}} = \max_{p(X)} I(X; Y | Z) = \max_{p(X)} (I(X; Y) - I(X; Z)), \quad (10.28)$$

where $p(X)$ is the probability distribution function of the transmitted signal.

Consider the case when the eavesdropper received the same information as the intended receiver, i.e., $Z = Y$. In that case, $I(X; Y | Z) = 0$ and hence, $C_s^{\text{DWTC}} = 0$. Physically, this implies that information-theoretic security cannot be achieved over a noiseless channel ($Z = Y$). For noiseless channels, secret keys must be used for obtaining security. Note that C_s^{DWTC} is the difference between the rate of information conveyed to the legitimate receiver vis-à-vis the rate of information leaked to the eavesdropper.

Definition 10.13 A Discrete Memoryless Channel (DMC) is **Weakly Symmetric** if the rows of the channel transition probability matrix are permutations of each other and the column sums $\sum_{x_i} p(y | x_i)$ are independent of y .

Example 10.2 Consider the binary symmetric channel shown in Fig. 10.7.

The channel transition probabilities are given by

$$P(Y | X) = \begin{bmatrix} 1-p & p \\ p & 1-p \end{bmatrix}. \quad (10.29)$$

It is easy to verify that the rows of the channel transition probability matrix are permutations of each other and the column sums $\sum_{x_j} p(y_i | x_j)$ are independent of y . Hence, the BSC is a weakly symmetric channel.

On the other hand, consider the binary erasure channel given by

$$P(Y | X) = \begin{bmatrix} 1-\varepsilon & \varepsilon & 0 \\ 0 & \varepsilon & 1-\varepsilon \end{bmatrix} \quad (10.30)$$

Here, the rows of the channel transition probability matrix are permutations of each other. However, the column sums $\sum_{x_j} p(y_i | x_j)$ are not independent of y . Hence, the binary erasure channel is not a weakly symmetric channel.

There are two interesting properties for weakly symmetric channels:

- *The capacity-achieving input distribution of a weakly symmetric channel is the uniform distribution over X .*
- *If the main channel and the eavesdropper's channel of a DWTC are both weakly symmetric, then*



$$C_s^{\text{DWTC}} = C_m - C_e, \quad (10.31)$$

where C_m is the capacity of the main channel and C_e is the capacity of the eavesdropper's channel.

Example 10.3 Consider the DWTC where the main channel and the tapped channel are both binary symmetric, as shown in Fig. 10.8.

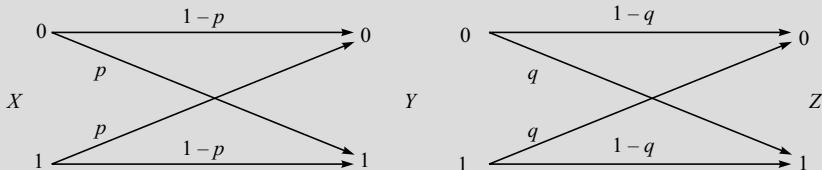


Fig. 10.8 A DWTC with weakly symmetric channels.

The capacity of the main channel is given by $C_m = 1 - H(p)$, where $H(p)$ is the entropy of a BSC (see Chapter 2). Similarly, the capacity of the eavesdropper channel is given by $C_e = 1 - H(p + q - 2pq)$. Recall that the eavesdropper channel is the channel from the source to the eavesdropper. Thus, the secrecy capacity of the DWTC is given by

$$C_s^{\text{DWTC}} = C_m - C_e = H(p + q - 2pq) - H(p) \quad (10.32)$$

The secrecy capacity C_s^{DWTC} as a function of p and q is plotted in Fig. 10.9. It is interesting to note that $p = 0.5$ will render the $C_s^{\text{DWTC}} = 0$, regardless of the value of q . This is because for $p = 0.5$, the capacity of the main channel goes to zero. For a fixed value of p (not equal to 0.5), as we worsen the eavesdropper channel (making q tend to 0.5 from either side), the C_s^{DWTC} improves.

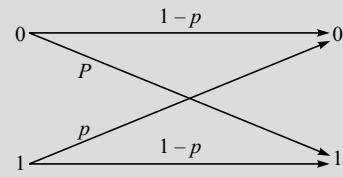


Fig. 10.7 A binary symmetric channel.

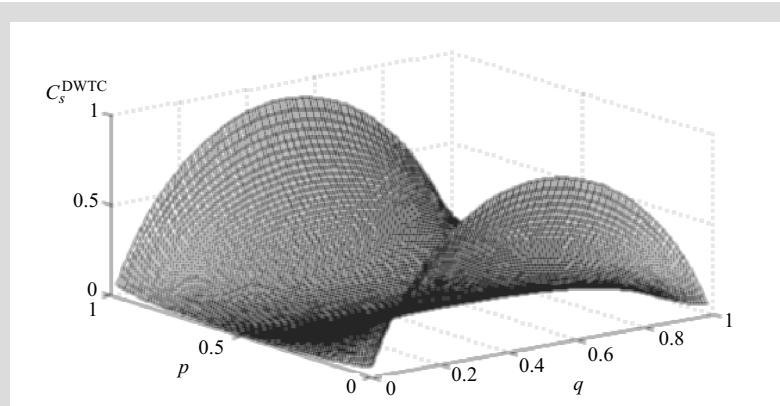


Fig. 10.9 The secrecy capacity C_s^{DWTC} as a function of p and q .

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 1:

- Consider a coding scheme where Alice sends a message $m \in M$ using a key $k \in K$ such that the codeword $z = m \oplus k$ where \oplus is the XOR operation and $z \in Z$. If K is independent of M , find the condition so that there is no leakage of information to the eavesdropper, i.e., $I(M; Z) = 0$.

S

- Again, consider a coding scheme where Alice sends a message $m \in M$ using a key $k \in K$ such that the codeword $z = m \oplus k$ where \oplus is the XOR operation and $z \in Z$. The message, the key and the codeword are all n -bit long. Let us assume that the two ‘easy-keys’, all-zeroes (00...0) and all-ones (11...1) have been removed from the set of valid keys and all the remaining keys are equi-probable. Determine whether this coding scheme satisfies the weak secrecy condition.

M

- Consider a DWTC with the main channel characterised by $\mathbf{P}_{Y|X} = \begin{bmatrix} 0.3 & 0.2 & 0.5 \\ 0.5 & 0.3 & 0.2 \\ 0.2 & 0.5 & 0.3 \end{bmatrix}$.

M

Find the secrecy capacity of the DWTC when the tapped channel is given by

$$\mathbf{P}_{Z|Y} = \begin{bmatrix} 0.1 & 0.3 & 0.6 \\ 0.6 & 0.1 & 0.3 \\ 0.3 & 0.6 & 0.1 \end{bmatrix}.$$

- Consider a DWTC with the main channel characterised by

$$\mathbf{P}_{Y|X} = \begin{bmatrix} \frac{1-p}{2} & \frac{1-p}{2} & \frac{p}{2} & \frac{p}{2} \\ \frac{p}{2} & \frac{p}{2} & \frac{1-p}{2} & \frac{1-p}{2} \end{bmatrix}. \text{ Find the secrecy capacity of the DWTC when the}$$

tapped channel is given by

$$(i) \quad P_{Z|Y} = \begin{bmatrix} 1-q & q & 0 & 0 \\ q & 1-q & 0 & 0 \\ 0 & 0 & 1-q & q \\ 0 & 0 & q & 1-q \end{bmatrix}.$$

$$(ii) \quad P_{Z|Y} = \begin{bmatrix} \frac{q}{2} & \frac{1-q}{2} & \frac{r}{2} & \frac{1-r}{2} \\ \frac{1-q}{2} & \frac{q}{2} & \frac{1-r}{2} & \frac{r}{2} \\ \frac{r}{2} & \frac{1-r}{2} & \frac{q}{2} & \frac{1-q}{2} \\ \frac{1-r}{2} & \frac{r}{2} & \frac{1-q}{2} & \frac{q}{2} \end{bmatrix}.$$

If you have successfully solved the above problems,
you have mastered LO 1. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/628>



10.4 The Gaussian Wiretap Model

LO 2



Describe the Gaussian wiretap model, calculate the secrecy capacity of wireless channels, define outage capacity and the outage probability.

Cheong and Hellman extended the Wyner's wiretap model to Gaussian channels. Additive white-Gaussian noise channels are commonly encountered in wireless communications. Following assumptions are made.

- The noise processes over the main and wire-tap channels are independent and identically distributed (i.i.d.) Gaussian over different channel uses.
- The noise processes over the main and wire-tap channels have zero mean and variances σ_1^2 and σ_2^2 , respectively.
- The average power constraint for the transmitted symbols is P .

It can be shown that the maximum achievable secure rate of communication, C_s , with a weak secrecy constraint, for the Gaussian wire-tap channel is

Levels of Difficulty

S Simple: Level 1 and Level 2 Category

M Medium: Level 3 and Level 4 Category

D Difficult: Level 5 and Level 6 Category

$$C_s = (C_m - C_e)^+, \quad (10.33)$$

where C_m is the capacity of the main channel, C_e is the capacity of the eavesdropper's channel and $(x)^+ = \max(x, 0)$. Once again, the eavesdropper's channel is a concatenation of the main and wiretap channels (Fig. 10.3). Using the expressions for the capacity of Gaussian channels, one can write the expression for the secrecy capacity of Gaussian wiretap channel as



$$C_s = \left(\frac{1}{2} \log \left(1 + \frac{P}{\sigma_1^2} \right) - \frac{1}{2} \log \left(1 + \frac{P}{\sigma_1^2 + \sigma_2^2} \right) \right)^+ \quad (10.34)$$

From (10.34), we observe that in order to have a positive secrecy capacity, we need to simply have $\sigma_2 > 0$ for any power P . That is, as long as the eavesdropper's channel is noisier (degraded) than the main channel we can have $C_s > 0$. This assumption is reasonable for the wire-tap model, which is a good model for wireline systems. However, this scenario may not be always true for wireless systems. In certain cases, the eavesdropper may actually be closer to the source than the legitimate receiver.

Example 10.4 Consider a Gaussian wiretap channel, whose secrecy capacity is given by (10.34).

For large P , the secrecy capacity is upper-bounded by

$$C_s|_{UB} = \left(\frac{1}{2} \log \left(\frac{\sigma_1^2 + \sigma_2^2}{\sigma_1^2} \right) \right)^+ \quad (10.35)$$

This upper-bound is plotted for different values of s_1 and s_2 in Fig. 10.10. If we represent the variance of the main channel as $\sigma_m^2 = \sigma_1^2$ and that of the eavesdropper channel as $\sigma_e^2 = \sigma_1^2 + \sigma_2^2$, we can write (10.35) as

$$C_s|_{UB} = \left(\frac{1}{2} \log \left(\frac{\sigma_e^2}{\sigma_m^2} \right) \right)^+ \quad (10.36)$$

It is interesting to note that, as P increases, the secrecy capacity does not increase unbounded, like channel capacity.

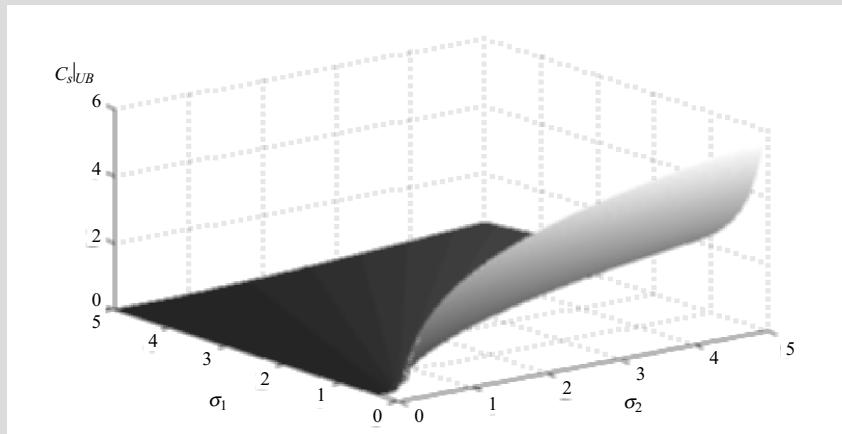


Fig. 10.10 The upper bound on the secrecy capacity for a Gaussian wiretap channel at high SNR.

10.5 Secrecy Capacity in Wireless Channels

LO 2

Although, the DWTC is interesting mathematically, it is not a practical model for wireless communication scenarios. In wireless communications, the eavesdropper can simply ‘stick-up’ the antenna and ‘listen in’ without the trouble of ‘tapping the wire’. Thus, there will be a direct wireless channel to the legitimate receiver (main channel), and another wireless channel to the eavesdropper (eavesdropper channel) as shown in Fig. 10.11.

Another interesting thing about the wireless communication scenario is that both the legitimate receiver and the eavesdropper are free to move around, and can be closer to the transmitter with respect to the other. Since the ‘channel quality’ depends on the distance between the transmitter-receiver pair, the relative qualities of the main channel and the eavesdropper channel can vary.

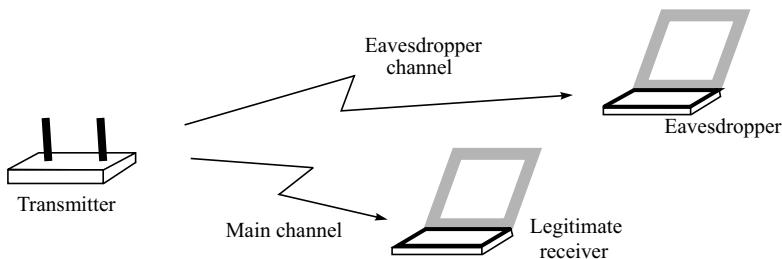


Fig. 10.11 A wireless scenario.

Definition 10.14 A channel $X-Z$ (characterised by $P_{Z|X}$) is **Noisier** than channel $X-Y$ (characterised by $P_{Y|X}$) if for every random variable U satisfying the Markov chain $U \rightarrow X \rightarrow YZ$,

$$I(U; Y) \geq I(U; Z)$$

A channel characterised by $P_{Y|X}$ is said to be **more capable** than $P_{Z|X}$ if $I(X; Y) \geq I(X; Z)$ for all inputs X . Less noisy implies more capable!

It can be shown that if the main channel is more capable than the eavesdropper’s channel, the secrecy capacity is given by

$$C_s = \max_{P_X} I(X; Y) - I(X; Z) \quad (10.37)$$

We note that both $I(X; Y)$ and $I(X; Z)$ are concave functions in the input distribution P_X . Hence, the difference $I(X; Y) - I(X; Z)$ may be either convex or concave in the input distribution P_X . Dijk has shown that in the event $P_{Y|X}$ is less noisy than $P_{Z|X}$, $I(X; Y) - I(X; Z)$ is a concave function in the input distribution P_X . If the eavesdropper’s channel is noisier than the main channel and both channels are weakly symmetric then the secrecy capacity is given by

$$C_s = C_m - C_e \quad (10.38)$$

where C_m is the capacity of the main channel and C_e is the capacity of the eavesdropper’s channel. If the main channel is noisier than the eavesdropper’s channel, the secrecy capacity is zero.



Example 10.5 Consider the wireless communication scenario where the main channel is a BSC(p) and the eavesdropper's channel is a BSC(q), as depicted in Fig. 10.12.

Assuming $p < q < 0.5$, the main channel is less noisy than the eavesdropper's channel. Also, both channels are weakly symmetric. Hence, the secrecy capacity is given by

$$\begin{aligned} C_s &= \max_X I(X;Y) - I(X;Z) \\ &= 1 - H(p) - (1 - H(q)) \\ &= H(q) - H(p) \end{aligned}$$

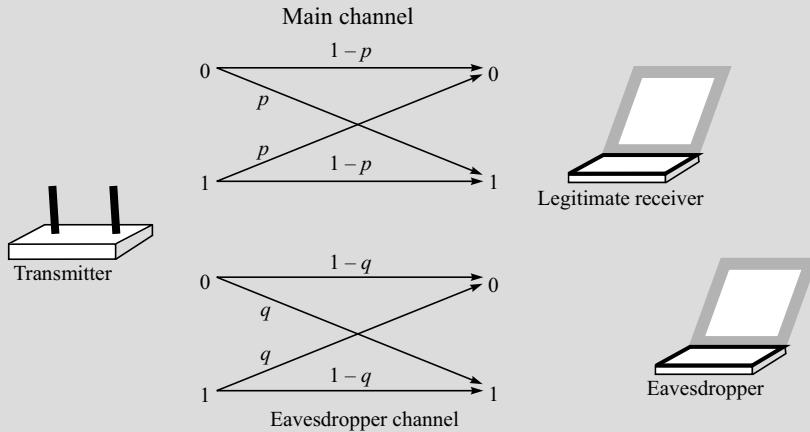


Fig. 10.12 A wireless scenario with BSCs.

Let us now consider a wireless communication scenario where there exists a flat fading channel between the transmitter and the legitimate receiver as well as the transmitter and the eavesdropper. The received samples y_k and z_k can be expressed as,

$$y_k = h_k x_k + n_k \quad (10.39)$$

and

$$z_k = g_k x_k + e_k \quad (10.40)$$

where h_k and g_k are the time-varying fading coefficients of the main channel and eavesdropper channel, respectively. The noise samples are represented by n_k and e_k , which are assumed to be complex AWGN with variance σ_m^2 and σ_e^2 respectively.

The average SNR at the legitimate receiver is given by $\text{SNR}_r = \frac{E[|h_k|^2]P}{\sigma_m^2}$ and the average SNR at the eavesdropper is given by $\text{SNR}_e = \frac{E[|g_k|^2]P}{\sigma_e^2}$.

The capacity of the main channel is

$$C_m = \frac{1}{2} \log \left(1 + \frac{|h_k|^2 P}{\sigma_m^2} \right) \quad (10.41)$$

and the capacity of the eavesdropper channel is

$$C_e = \frac{1}{2} \log \left(1 + \frac{|g_k|^2 P}{\sigma_e^2} \right) \quad (10.42)$$

Therefore, the secrecy capacity can be expressed as

$$C_s = \left(\frac{1}{2} \log \left(1 + \frac{|h_k|^2 P}{\sigma_m^2} \right) - \frac{1}{2} \log \left(1 + \frac{|g_k|^2 P}{\sigma_e^2} \right) \right)^+ \quad (10.43)$$

Intuition Since h_k and g_k are random variables, C_m and C_e are random variables. Consequently, C_s is also a random variable. This motivates us to characterise the secrecy of the system in terms of outage. Sometimes, a **Block Fading Model** is assumed, which implies that the channel gain matrices H_k and G_k remain constant over a *block* of large number of symbols, and the gains are assumed to be independent across blocks. Can we use the random nature of the fading and carry out opportunistic use of the fading channel for secure communications? The following example explores that possibility.

Example 10.6 Consider a wireless communication scenario with a source, a legitimate receiver and an eavesdropper. Let h_k and g_k be the time-varying fading coefficients of the main channel and eavesdropper channel, and the channels are affected by complex AWGN with variance σ_m^2 and σ_e^2 respectively. From (10.43), whenever $\frac{|h_k|^2}{\sigma_m^2} \geq \frac{|g_k|^2}{\sigma_e^2}$, $C_s \geq 0$, the transmitter could use the opportunity and transmit (secure timeslot, S). Otherwise, it should not transmit (insecure timeslot, I). This opportunistic strategy is captured in Fig. 10.13. The bold line represents the SNR of the legitimate receiver while the dotted line represents the SNR of the eavesdropper.

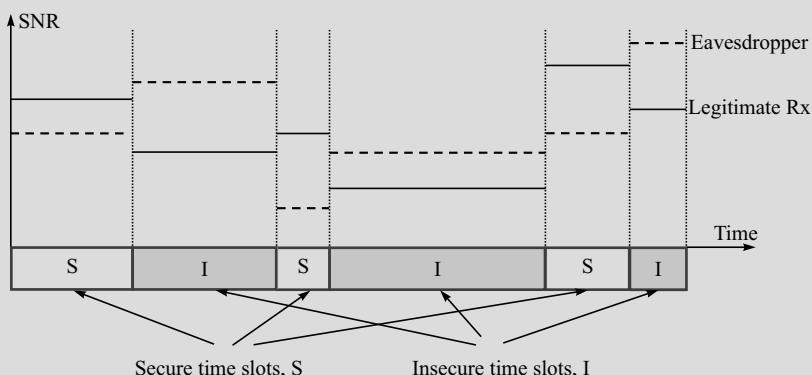


Fig. 10.13 Opportunistic use of the fading channel for secure communications.

Definition 10.15 An **Outage** occurs if the secrecy capacity is smaller than a certain fixed value, called **Outage Capacity**. The **Outage Probability** for a certain outage capacity C_{outage} is defined as $\Pr\{C_s < C_{\text{outage}}\}$.

Example 10.7 Consider a wireless communication scenario with a source, a legitimate receiver and an eavesdropper. Since it is a wireless scenario, the eavesdropper is free to move. Let the distance between the transmitter and the intended receiver be fixed with the signal to noise ratio of the main channel, $\text{SNR}_m = 20 \text{ dB}$. Suppose the eavesdropper is farther away from the transmitter as compared to the intended receiver, and the signal to noise ratios of the eavesdropper channel, $\text{SNR}_e = 10 \text{ dB}$. Figure 10.14 shows the variation of the outage probability versus outage capacity. When $\text{SNR}_e = 10 \text{ dB}$ (i.e., the eavesdropper channel is degraded with respect to the main channel), we get the outage probabilities varying in the range of $0.1 < P_{\text{outage}} < 1$ for different outage capacities. Suppose the eavesdropper moves to a location that is at a similar distance as the intended receiver, leading to $\text{SNR}_m = \text{SNR}_e = 20 \text{ dB}$. In this case, we get the outage probabilities varying in the range of $0.5 < P_{\text{outage}} < 1$.

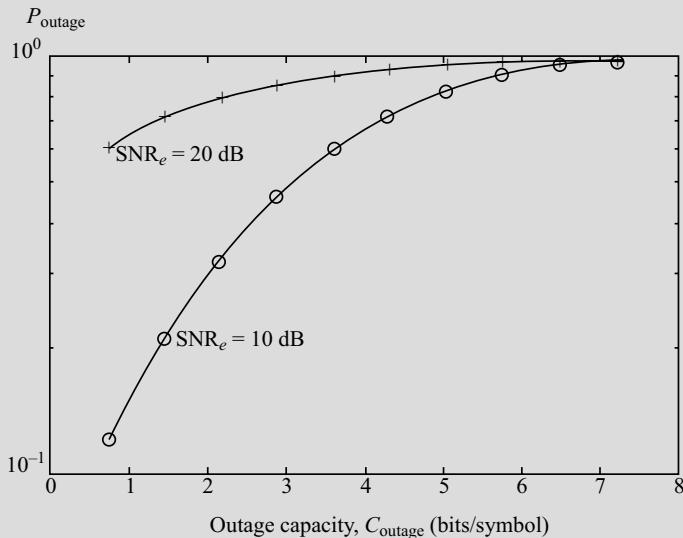


Fig. 10.14 Outage probability versus outage capacity for a fixed $\text{SNR}_m = 20 \text{ dB}$.

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 2:

- Consider the wireless communication scenario where the main channel is a Binary Erasure Channel (BEC(ε_1)) and the eavesdropper channel is a BEC(ε_2) with $\varepsilon_2 > \varepsilon_1$. Find the secrecy capacity.
- Consider the relay-based communication scenario, shown in Fig. 10.15, where the relay channel is a BSC(r), the main channel is a BSC(p) and the eavesdropper's channel is a BSC(q). Find the secrecy capacity.

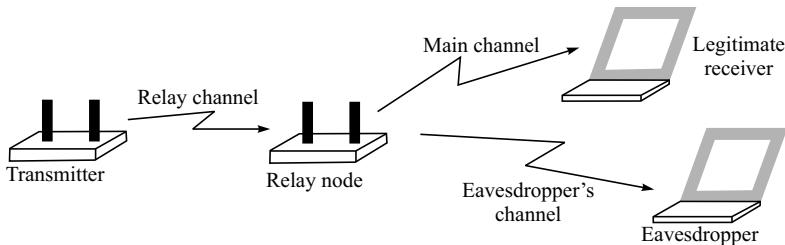


Fig. 10.15

- Consider a wireless communication scenario with a source, a legitimate receiver and an eavesdropper. For $\text{SNR}_m = \text{SNR}_e = 20$ dB the $C_{\text{outage}} = 0.8$ bits/symbol and the $P_{\text{outage}} = 0.6$. What will be the increase in C_{outage} if SNR_e falls to 10 dB?
- Again consider a wireless communication scenario with a source, a legitimate receiver and an eavesdropper. For $\text{SNR}_m = \text{SNR}_e = 20$ dB the $P_{\text{outage}} \approx 0.7$ bits/symbol for a corresponding $C_{\text{outage}} \approx 1.4$. What will be the improvement in P_{outage} if SNR_e falls to 10 dB?
- In a wireless communication scenario, the receiver and the eavesdropper are equidistant from the source, i.e., $\text{SNR}_m = \text{SNR}_e$. If, h_k and g_k are the time-varying fading coefficients of the main channel and eavesdropper channel.
 - What will be the secrecy capacity, C_s , when $|h_k| < |g_k|$?
 - What will be the $P(C_s = 0)$?

If you have successfully solved the above problems,
you have mastered LO 2. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrpage.flipick.com/index.php/629>



10.6 Cooperative Jamming

So far we have developed the intuition that ‘the higher the degradation of the eavesdropper’s channel, the better is the secrecy capacity’. This motivates us to consider ways and means to introduce ‘interference or noise’ in the eavesdropper channel without affecting the main channel. The pertinent question is how we can degrade the eavesdropper channel without degrading the main channel? This has led to the idea of cooperative jamming. Traditionally, the term ‘jamming’ refers to intentional prevention of radio communications using electromagnetic signals. Jamming was first used in military communications during the Second World War, and is still used in modern communications. However, cooperative jamming is a relatively new term, and is a fairly effective tool, as we will discover in this section.

LO 3

Explain the idea behind cooperative jamming, artificial noise forwarding and solve problems related to friendly jammer with single and multiple antennas.

Example 10.8 Consider a wireless communication scenario as shown in Fig. 10.16 with a transmitter, T, a legitimate receiver, R, an eavesdropper, E, and a friendly jammer, J. Z_1 and Z_2 are zero mean Gaussian random variables with variances σ_m^2 and σ_e^2 respectively. Let the friendly jammer transmit a Gaussian i.i.d. sequence with variance σ_J^2 . We assume that this friendly jammer is close to the eavesdropper and far away from the legitimate receiver. Thus, it contributes to degrading the channel of the eavesdropper only (is this a fair assumption?). In the absence of the friendly jammer, the secrecy capacity is given by $C_s = (C_m - C_e)^+$, where C_m is the capacity of the main channel and C_e is the capacity of the eavesdropper’s channel. In the presence of the friendly jammer, the secrecy capacity of T can be written as

$$C_s = \left(\frac{1}{2} \log \left(1 + \frac{P}{\sigma_m^2} \right) - \frac{1}{2} \log \left(1 + \frac{P}{\sigma_e^2 + \sigma_J^2} \right) \right)^+ \quad (10.44)$$

For large P , the secrecy capacity is upper bounded by

$$C_s|_{UB} = \left(\frac{1}{2} \log \left(\frac{\sigma_e^2 + \sigma_J^2}{\sigma_m^2} \right) \right)^+ \quad (10.45)$$

Comparing it with (10.36), we note that cooperative jamming improves the secrecy rate.

DID YOU KNOW ? Cooperative jamming can be carried out in several ways: cooperative jamming with noise, cooperative jamming with a random code and cooperative jamming with a structured code.

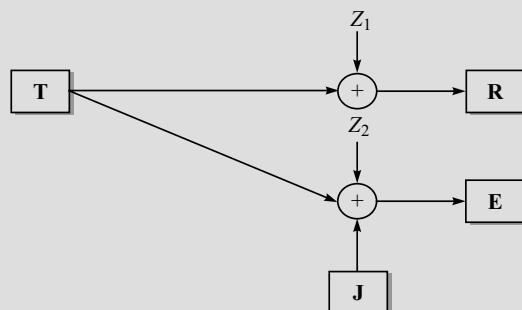


Fig. 10.16 A wiretap channel with a friendly jammer.

Suppose the friendly jammer has multiple antennas. In that case, it can choose the weight vector, w , in such a manner that it directs a beam towards the eavesdropper and a null towards the legitimate receiver. This scenario is illustrated in Fig. 10.17. In order to achieve this, the friendly jammer must have the channel

state information. That is, the channel between the jammer and eavesdropper, \mathbf{h}_{JE} and the channel between the jammer and legitimate receiver, \mathbf{h}_{JR} . The friendly jammer can then use beam forming techniques to create high gain in the direction of the eavesdropper while creating a null in the direction of the receiver. Mathematically, the jammer must do the following.

$$\begin{aligned} & \text{maximize } |\mathbf{w}'\mathbf{h}_{JE}|^2 \\ & \text{subject to } \begin{cases} \mathbf{w}'\mathbf{h}_{JR} = 0 \\ \|\mathbf{w}\|^2 \leq P_J \end{cases} \end{aligned} \quad (10.46)$$

where P_J is the power available to the jammer. Here, the transmitter relies upon an external jammer to selectively degrade the channel of the eavesdropper. Is it possible to selectively degrade the channel of the eavesdropper without the use of cooperative jammer? We explore this possibility in the next section.

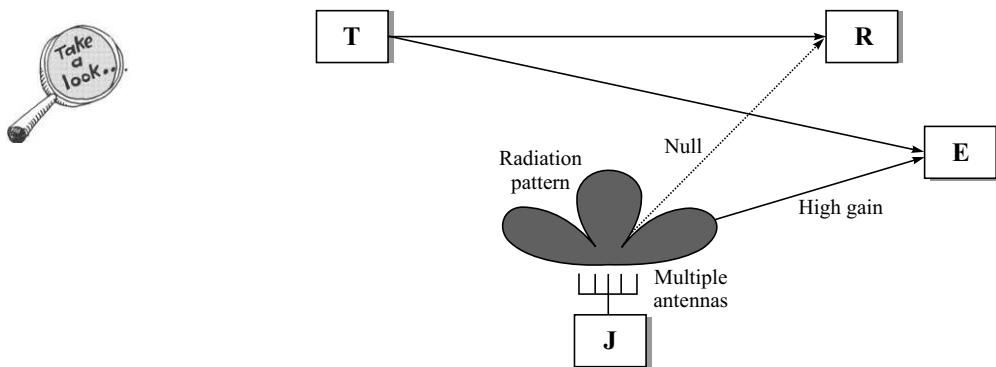


Fig. 10.17 A friendly jammer with multiple antennas.

10.7 Artificial Noise Forwarding

In the previous section we studied the use of a cooperative node that acts as a friendly jammer and degrades the channel of the eavesdropper. The friendly jammer needs to know the channel between the jammer and the eavesdropper. However, availability of a friendly jammer may not be always possible. Is it possible for the transmitter to degrade the signal received by the eavesdropper without affecting the legitimate user? This is possible by way of noise forwarding. Noise forwarding is a technique in which the transmitter transmits an artificially generated noise along with the information signal. The artificial noise is generated in such a manner that it does not affect the received signal at the legitimate receiver. We assume that the transmitter has multiple antennas, while the receiver and the eavesdropper have a single antenna element each, as depicted in Fig. 10.18. Here, \mathbf{h}_{TR} is the channel between the transmitter and the legitimate receiver and \mathbf{h}_{TE} is the

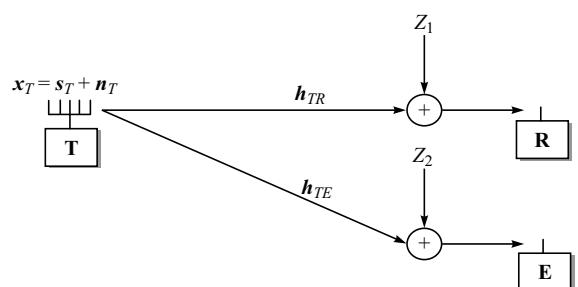


Fig. 10.18 A wireless scenario with multiple antennas at the transmitter.

channel between the transmitter and the eavesdropper. Z_1 and Z_2 are zero mean Gaussian random variables with variances σ_m^2 and σ_e^2 respectively.

Let the transmitter send a sum of two signals: information signal and artificial noise as follows.

$$\mathbf{x}_T = \mathbf{s}_T + \mathbf{n}_T \quad (10.47)$$

where \mathbf{s}_T represents the information signal and \mathbf{n}_T represents the artificial noise. In general, both \mathbf{s}_T and \mathbf{n}_T can be complex Gaussian vectors. By design, $\mathbf{h}_{TR}\mathbf{n}_T = 0$. We assume that the transmitter has a power constraint, i.e., $E[\mathbf{x}_T' \mathbf{x}_T] \leq P_0$. From (10.47) it is clear that the transmitter divides the total available power between information signal and the artificial noise. Thus, the power in the information symbol is $E[\mathbf{s}_T' \mathbf{s}_T] = P_{inf} < P_0$. This results in a lower information rate. At the legitimate receiver, the received signal is given by

$$y_R = \mathbf{h}_{TR}\mathbf{s}_T + \mathbf{h}_{TR}\mathbf{n}_T + Z_1 = \mathbf{h}_{TR}\mathbf{s}_T + Z_1, \quad (10.48)$$

where Z_1 represents the noise at the legitimate receiver. Note that, we have put $\mathbf{h}_{TR}\mathbf{n}_T = 0$. At the eavesdropper, the received signal is given by

$$y_E = \mathbf{h}_{TE}\mathbf{s}_T + \mathbf{h}_{TE}\mathbf{n}_T + Z_2, \quad (10.49)$$

where Z_2 represents the noise at the eavesdropper. The effective noise for the eavesdropper is $\mathbf{h}_{TE}\mathbf{n}_T + Z_2$ and the total noise power is given by $E|\mathbf{h}_{TE}\mathbf{n}_T|^2 + \sigma_e^2$. The secrecy capacity of T can be written as

$$C_s = \left(\frac{1}{2} \log \left(1 + \frac{E|\mathbf{h}_{TR}\mathbf{s}_T|^2}{\sigma_m^2} \right) - \frac{1}{2} \log \left(1 + \frac{E|\mathbf{h}_{TE}\mathbf{s}_T|^2}{\sigma_e^2 + E|\mathbf{h}_{TE}\mathbf{n}_T|^2} \right) \right)^+ \quad (10.50)$$

We observe the following.

- (i) In the worst case, the eavesdropper is located ‘very close’ to the transmitter, i.e., $\sigma_e^2 \rightarrow 0$. This yields the minimum guaranteed secrecy capacity (regardless of the location of the eavesdropper).
- (ii) In the worst case scenario ($\sigma_e^2 \rightarrow 0$) if there is no artificial noise injection ($\mathbf{n}_T \rightarrow 0$), the second term in (10.50) tends to infinity, thereby making $C_s \rightarrow 0$.

INDUSTRIAL RELEVANCE



Example 10.9

The broadcast nature of the wireless medium makes it hard to eliminate unauthorised access to wireless networks. Different types of attacks encountered in the wireless industry today are given in Table 10.1. Physical layer security addresses an important type of passive attack—eavesdropping. Techniques, such as artificial noise injection and friendly jamming will help secure wireless communication at the physical layer.

Table 10.1

Active attacks	Passive attacks
Denial of service attack	Eavesdropping
Resource consumption	Traffic analysis
Replay attack	
Massage modification	

Learning Review

YOU ARE NOW READY TO ATTEMPT THE FOLLOWING QUESTIONS LINKED TO LO 3:

- Consider secrecy using artificial noise forwarding where the secrecy capacity of transmitter can be written as $C_s = \left(\frac{1}{2} \log \left(1 + \frac{E|\mathbf{h}_{TR} s_T|^2}{\sigma_m^2} \right) - \frac{1}{2} \log \left(1 + \frac{E|\mathbf{h}_{TE} s_T|^2}{\sigma_e^2 + E|\mathbf{h}_{TE} \mathbf{n}_T|^2} \right) \right)^+$. What will be the effect on secrecy if we increase the number of transmit antennas? Explain.
- In a wireless scenario, we usually do not have control over the location of the eavesdropper. What is the worst-case secrecy capacity if we are using artificial noise forwarding and permit the eavesdropper to be present at any location?
- Consider a wireless communication scenario as shown in Fig. 10.19 with a transmitter, T, a legitimate receiver, R, an eavesdropper, E, and a friendly jammer, J. Z_1 and Z_2 are zero mean Gaussian random variables with variances σ_m^2 and σ_e^2 respectively. Find the secrecy capacity in the presence of the friendly jammer. For what values of α will $C_s = 0$?

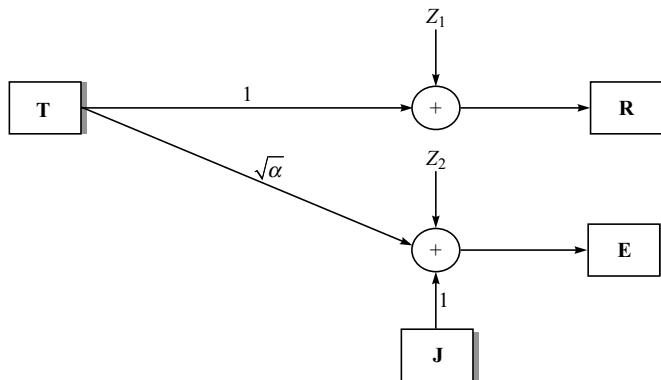


Fig. 10.19

If you have successfully solved the above problems,
you have mastered LO 3. Congratulations!



The answers are available here. Scan to check

Or

Visit <http://qrcode.flipick.com/index.php/630>



10.8 Concluding Remarks

The notion of one-time pads was proposed by Vernam in early 1926 and was used during the First World War. Shannon, in his seminal paper published in 1949, titled ‘Communication Theory of Secrecy Systems,’ put forth the initial ideas for information-theoretic analysis of secrecy. The DWTC was first studied by Wyner in 1975. He also proposed the idea of secrecy capacity. Later in 1978, Csiszár (pronounced Chee_sar) and Körner extended Wyner’s work to general broadcast scenarios. The secrecy capacity of the degraded Gaussian wiretap channel was studied by Cheong and Hellman in 1978. At this time, MIMO systems were generally unknown. After the advent of MIMO systems, information-theoretic analysis for MIMO systems was carried out. In 2007, Khisti and Wornell and Liu and Shamai were original contributors to this field. Fading, which is a bane for wireless communications, has a positive role to play for secrecy. This was first highlighted by Barros and Rodrigues in 2006. The notion of cooperative jamming was proposed by He and Yener in 2008. The idea of adding artificial noise to enhance secrecy was introduced by Goel and Negi in 2008. The area of information-theoretic secrecy is gaining maturity. Currently, the research community in this area is engaged in constructing practical codes for secrecy and exploring communications and signal processing issues in physical-layer security.

LEARNING OUTCOMES

- The communication over the main channel is perfectly secure if and only if the entropy of the message M , given the eavesdropper’s observation Z , is equal to the entropy of the original message M , i.e., $H(M|Z) = H(M)$.
- Strong secrecy implies that the message and the eavesdropper’s observations are almost independent while weak secrecy implies that the normalised mutual information between the message and the eavesdropper vanishes, i.e.,

$$\lim_{n \rightarrow \infty} I(M; Z^n) \rightarrow 0 \quad (\text{Strong Secrecy})$$

$$\lim_{n \rightarrow \infty} \frac{1}{n} I(M; Z^n) \rightarrow 0 \quad (\text{Weak Secrecy})$$

- The eavesdropper’s channel, which is the composite of the main channel and the wiretap channel, is simply a degraded version of the main channel. This type of channel is called the degraded wiretap channel (DWTC).
- A DWTC consists of the following:
 - (i) a random source at the encoder
 - (ii) a message set M of size 2^{nR}
 - (iii) an encoding function, f , which maps a message $m \in M$ to the codeword X^n , i.e., $m \rightarrow X^n$.
 - (iv) a decoding function, g , which maps the channel observation Z^n to a message, i.e., $Z^n \rightarrow \hat{m}$.
- The equivocation, which is a measure of the confusion created at the eavesdropper’s end, is given by $H(M|Z^n)$. If a code, C_n , with block length n is used, we can write the equivocation as $E(C_n) = H(M|Z^n C_n)$.
- The equivocation rate is the normalised equivocation, expressed as $\Delta = \frac{H(M|Z^n)}{H(M)}$.

- The leakage of information to the eavesdropper is expressed as $L(C_n) = I(M; Z^n | C_n)$.
- A weak rate-equivocation pair (R, R_e) is achievable for DWTC, if there exists a sequence of $(2^{nR}, n)$ codes $\{C_n\}$ such that

$$\lim_{n \rightarrow \infty} P_e(C_n) = 0$$

$$\lim_{n \rightarrow \infty} \frac{1}{n} E(C_n) \geq R_e$$

- Weak rate-equivocation region for DWTC is given by

$$\mathcal{R} = \text{cl}(\{(R, R_e) : (R, R_e) \text{ is achievable}\})$$

where $\text{cl}(X)$ refers to the closure of the set X .

- For any distribution p_X on X , we define the set $\mathcal{R}^{\text{DWTC}}(p_X)$ as

$$\mathcal{R}^{\text{DWTC}}(p_X) = \left\{ (R, R_e) : \begin{array}{l} 0 \leq R_e \leq R \leq I(X; Y) \\ 0 \leq R_e \leq I(X; Y|Z) \end{array} \right\},$$

where the joint distribution of X, Y, Z factorises as $p_X p_{Y|X} p_{Z|Y}$. Then the rate-equivocation region of the DWTC is the convex region given by

$$\mathcal{R}^{\text{DWTC}} = \bigcup_{p_X} \mathcal{R}^{\text{DWTC}}(p_X).$$

- If Rate-Equivocation Pair (R, R_e) is achievable with $R = R_e$ then R is Full Secrecy Rate, i.e., $R_e = \lim_{n \rightarrow \infty} \frac{1}{n} H(M | Z^n) = \lim_{n \rightarrow \infty} \frac{1}{n} H(M) = R$.
- The secrecy capacity is the maximum possible transmission rate for which the eavesdropper is unable to decode any information. Here, the eavesdropper is assumed to have unlimited computing resources and time, and hence, the secrecy is *provable*.
- The strong secrecy capacity of a DWTC is given by

$$C_s^{\text{DWTC}} = \sup_R \{R : (R, R) \in \mathcal{R}^{\text{DWTC}}\}.$$

- The secrecy capacity of a DWTC is given by

$$C_s^{\text{DWTC}} = \max_{p(X)} I(X; Y | Z) = \max_{p(X)} I(X; Y) - I(X; Z)$$

where $p(X)$ is the probability distribution function of the transmitted signal.

- A Discrete Memoryless Channel (DMC) is weakly symmetric if the rows of the channel transition probability matrix are permutations of each other and the column sums $\sum_{x_i} p(y | x_i)$ are independent of y .
- If the main channel and the eavesdropper's channel of a DWTC are both weakly symmetric, then $C_s^{\text{DWTC}} = C_m - C_e$, where C_m is the capacity of the main channel and C_e is the capacity of the eavesdropper's channel.
- The maximum achievable secure rate of communication, C_s , with a weak secrecy constraint, for the Gaussian wire-tap channel is $C_s = (C_m - C_e)^+$, where C_m is the capacity of the main channel and C_e is the capacity of the eavesdropper's channel.
- The secrecy capacity of Gaussian wiretap channel is

$$C_s = \left(\frac{1}{2} \log \left(1 + \frac{P}{\sigma_1^2} \right) - \frac{1}{2} \log \left(1 + \frac{P}{\sigma_1^2 + \sigma_2^2} \right) \right)^+, \text{ where } (x)^+ = \max(x, 0)$$

- For large transmitter power, P , the secrecy capacity of a Gaussian wiretap channel is upper bounded by $C_S|_{UB} = \left(\frac{1}{2} \log \left(\frac{\sigma_e^2}{\sigma_m^2} \right) \right)^+$, where σ_m^2 and σ_e^2 are the noise power of the main channel and the eavesdropper channel respectively.

- In a wireless communication scenario with flat fading, the secrecy capacity can be expressed as

$$C_s = \left(\frac{1}{2} \log \left(1 + \frac{|h_k|^2 P}{\sigma_m^2} \right) - \frac{1}{2} \log \left(1 + \frac{|g_k|^2 P}{\sigma_e^2} \right) \right)^+$$

where h_k and g_k are the time-varying fading coefficients of the main channel and eavesdropper channel, respectively, and σ_m^2 and σ_e^2 are the noise power of the main channel and the eavesdropper channel, respectively. Since h_k and g_k are random variables, C_s is a random variable.

- An outage occurs if the secrecy capacity is smaller than a certain fixed value, called outage capacity. The outage probability for a certain outage capacity C_{outage} is defined as $\Pr\{C_s < C_{\text{outage}}\}$.
- In the presence of the friendly jammer, the secrecy capacity of the transmitter is

$$C_s = \left(\frac{1}{2} \log \left(1 + \frac{P}{\sigma_m^2} \right) - \frac{1}{2} \log \left(1 + \frac{P}{\sigma_e^2 + \sigma_j^2} \right) \right)^+ \text{ where } \sigma_m^2, \sigma_e^2 \text{ and } \sigma_j^2 \text{ are the noise power of the main}$$

channel, the eavesdropper channel and the friendly jammer respectively.

An expert is a person who has made all the mistakes
that can be made in a very narrow field.

Niels Bohr (1885-1962)



MULTIPLE CHOICE QUESTIONS

- 10.1 In Shannon's notion of secrecy, he assumed

- (a) The main channel is noiseless
- (b) The eavesdropper's channel is noiseless
- (c) Both (a) and (b)
- (d) None of the above

- 10.2 In Shannon's notion of perfect secrecy, for original message M , and the eavesdropper's observation Z ,

- | | |
|---------------------|-----------------------|
| (a) $H(Z M) = H(M)$ | (b) $H(M Z) = H(M)$ |
| (c) $H(M Z) = H(Z)$ | (d) None of the above |

- 10.3 For original message M , and the eavesdropper's observation Z , the leakage of information to the eavesdropper is given by
- (a) $I(M; Z) - H(M|Z)$
 - (b) $H(M) + H(M|Z)$
 - (c) $I(M; Z)$
 - (d) None of the above
- 10.4 For original message M , shared secret key K and the eavesdropper's observation Z , perfectly secure communication requires
- (a) $H(K) \geq H(M)$
 - (b) $H(K) \leq H(M)$
 - (c) $H(K) \geq H(Z)$
 - (d) None of the above
- 10.5 In Wyner's wire-tap model, it is assumed that
- (a) The eavesdropper's channel is noisier than the main channel
 - (b) The eavesdropper receives a degraded version of the signal
 - (c) Both (a) and (b)
 - (d) None of the above
- 10.6 In Wyner's weak secrecy constraint, for blocklength n and the shared secret key K
- (a) $\lim_{n \rightarrow \infty} \frac{1}{n} I(K; Z^n) \rightarrow 0$
 - (b) $\lim_{n \rightarrow \infty} \frac{1}{n} I(M; K) \rightarrow 0$
 - (c) $\lim_{n \rightarrow \infty} \frac{1}{n} I(M; Z^n) \rightarrow 1$
 - (d) $\lim_{n \rightarrow \infty} \frac{1}{n} I(M; Z^n) \rightarrow 0$
- 10.7 For blocklength n , original message M , shared secret key K and the eavesdropper's observation Z , equivocation is given by
- (a) $I(M; Z^n)$
 - (b) $H(K|Z^n)$
 - (c) $H(M|Z^n)$
 - (d) $H(Z^n)$
- 10.8 The aim of secure communication over unreliable channels is
- (a) Reliable communication at a reasonable transmission rate, R
 - (b) Secret communication at a reasonable equivocation rate, Δ
 - (c) Both (a) and (b)
 - (d) None of the above
- 10.9 A weak rate-equivocation pair (R, R_e) is achievable for DWTC if there exists a sequence of $(2^{nR}, n)$ codes $\{C_n\}$ such that
- (a) $\lim_{n \rightarrow \infty} P_e(C_n) = 0$
 - (b) $\lim_{n \rightarrow \infty} \frac{1}{n} E(C_n) \geq R_e$
 - (c) Both (a) and (b)
 - (d) None of the above
- 10.10 In a Gaussian wiretap channel, it is assumed that the noise processes over the main and wire-tap channels are
- (a) Independent over different channel uses
 - (b) Identically distributed (i.i.d.) Gaussian over different channel uses
 - (c) Both (a) and (b)
 - (d) None of the above

For interactive quiz with answers, scan the QR code given here

Or

Visit <http://qrpage.flipick.com/index.php/575>



SHORT-ANSWER TYPE QUESTIONS



- 10.1 In physical layer security we assume that the eavesdropper has unlimited computational and time resources. True or false?
- 10.2 Give Shannon's definition of perfect secrecy.
- 10.3 For a given message M , and the eavesdropper's observation Z , we must have $I(M; Z) > 0$ for perfect secrecy. True or false?
- 10.4 What is the main difference in the underlying assumption between Shannon's secrecy model and Wyner's wire-tap model?
- 10.5 Suppose we have the binary erasure wiretap channel. What is the value of $I(M; Z^n)$?
- 10.6 Define equivocation rate.
- 10.7 The lower the equivocation rate, the more secure is the communication. True or false?
- 10.8 Define a weakly symmetric discrete memoryless channel.
- 10.9 What is the secrecy capacity of a Gaussian wiretap channel?
- 10.10 What is meant by full secrecy?
- 10.11 What is meant by secrecy capacity?
- 10.12 What is the secrecy capacity of a DWTC?
- 10.13 What is the secrecy capacity for a flat fading wireless channel with h_k and g_k as the time-varying fading coefficients of the main channel and eavesdropper channel, and σ_m^2 and σ_e^2 the noise power of the main channel and the eavesdropper channel respectively?
- 10.14 What is meant by outage?
- 10.15 What is the secrecy capacity in the presence of the friendly jammer given σ_m^2 , σ_e^2 and σ_j^2 are the noise power of the main channel, the eavesdropper channel and the friendly jammer respectively?

For answers, scan the QR code given here



Or

Visit <http://qrcode.flipick.com/index.php/632>

PROBLEMS

- M** 10.1 Show that, for perfect secrecy, the entropy of the key, K should be greater than or equal to the entropy of the message, M , i.e., $H(K) \geq H(M)$.
- M** 10.2 Suppose we perform a one-time pad for only the first $n-t$ bits of the binary message M using a key, K , with length $n-t$ bits. The remaining t bits of the message M are left unprotected. The resulting binary codeword X^n will be of length n . Let the eavesdropper intercept the codeword X^n directly. Show that for $t = \lfloor n \rfloor$:
 - (i) this scheme does not satisfy the strong secrecy criterion
 - (ii) this scheme satisfies the weak secrecy criterion.
- D** 10.3 Consider a coding scheme where Alice sends a message $m \in M$ using a key $k \in K$ such that the codeword $z = m \oplus k$ where \oplus is the XOR operation and $z \in Z$. The message, the key and the codeword

are all n -bit long binary strings. Suppose, the all-zero key is highly likely, with a probability $1/n$, while all other non-zero keys are equally likely. Show that this scheme:

- (i) does not satisfy the strong secrecy criterion
- (ii) satisfies the weak secrecy criterion.

s 10.4 Find the secrecy capacity of the DWTC with asymmetric Z channels shown in Fig. 10.20.

- (i) What is the C_s for $p = 0.1$?
- (ii) Plot C_s^{DWTC} and C_s versus p . Compare and comment on your result.

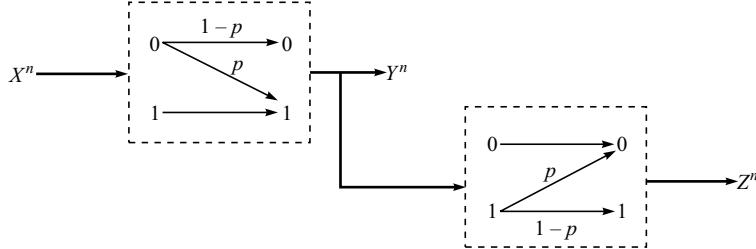


Fig. 10.20

s 10.5 Find the secrecy capacity of the DWTC with a BSC concatenated with a channel shown in Fig. 10.21.

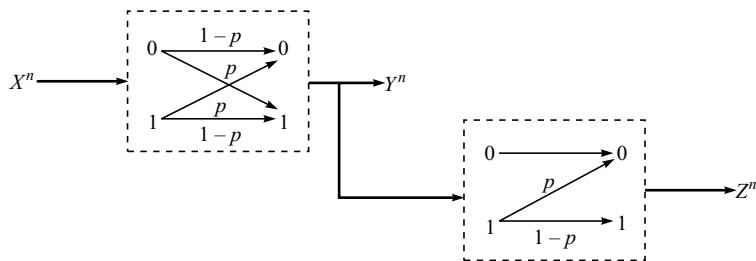


Fig. 10.21

s 10.6 Which of the following channels are weakly symmetric? Explain.

$$(i) P(Y | X) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$(ii) P(Y | X) = \begin{bmatrix} 1-p & p \\ q & 1-q \end{bmatrix}$$

$$(iii) P(Y | X) = \begin{bmatrix} 1-p-q & p & q \\ q & 1-p-q & p \\ p & q & 1-p-q \end{bmatrix}$$

$$(iv) P(Y | X) = \begin{bmatrix} 1-p-q & p & 0 & q & 0 \\ p & 0 & 1-p-q & 0 & q \\ q & 0 & p & 0 & 1-p-q \end{bmatrix}$$

- M** 10.7 Consider a broadcast scenario in wireless communications where the main channel is a BEC (ε) and the eavesdropper's channel is BSC (p). Find the condition that the eavesdropper's channel is noisier than the main channel.

- D** 10.8 Consider a DWTC with the main channel characterised by $P_{Y|X} = \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$. Find the

secrecy capacity of the DWTC when the tapped channel is given by $P_{Z|Y} = \begin{bmatrix} 0.1 & 0.8 & 0.1 \\ 0.1 & 0.1 & 0.8 \\ 0.8 & 0.1 & 0.1 \end{bmatrix}$. For

what choice of $P_{Z|Y}$ will the capacity of the eavesdropper's channel be half of the capacity of the main channel?

- D** 10.9 Consider the relay-based communication scenario, shown in Fig. 10.22, where the relay channel is a BSC(r) and the main channel is a BSC(p). The eavesdropper has positioned itself such that it can get direct transmission from the source (eavesdropper's channel 1, BSC(q_1)) as well as from the relay node (eavesdropper's channel 2, BSC(q_2)). Find the secrecy capacity if the eavesdropper simply uses the less noisy channel.

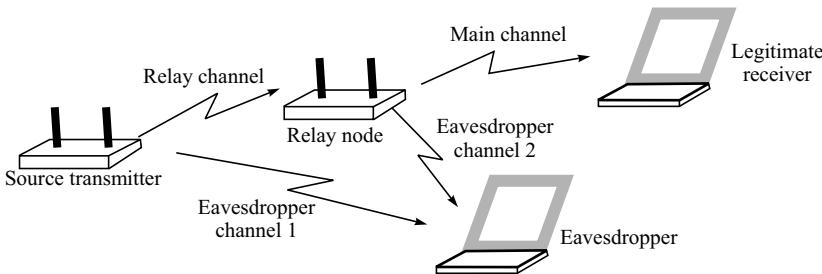


Fig. 10.22

- M** 10.10 Consider a wireless scenario where the transmitter has channel state information (CSI) about the main channel. Is fading useful from the perspective of secrecy capacity? Explain mathematically.

- M** 10.11 Consider a wireless scenario where there is a transmitter, T, a legitimate receiver, R, and an eavesdropper, E. Assume there is no fading. Let the distance between the transmitter-receiver be d_{TR} , transmitter- eavesdropper be d_{TE} . Model the received power $P_R = \frac{KP_T}{d^n}$ where P_T is the transmit power, d is the distance between the transmitter and receiver, n is the path loss exponent and K is a constant. Let the noise in the main channel and the eavesdropper channel be complex AWGN with variance σ_m^2 and σ_e^2 respectively. Derive an expression for the secrecy capacity of the transmitter. Does a higher value of the path loss exponent improve secrecy?

- M** 10.12 Suppose we have a linear array of Q equally-spaced friendly jammers between the transmitter and receiver. These jammers are located on the straight line joining the transmitter and receiver. Let the distance between the transmitter and receiver be D , and the eavesdropper be located exactly midway between the transmitter and receiver. Model the received power $P_R = \frac{KP_T}{d^n}$ where P_T

is the transmit power, d is the distance between the transmitter and receiver, n is the path loss exponent and K is a constant. Suppose the noise in all channels is complex AWGN with variance σ_n^2 . Let the total power available to the transmitter and all the friendly jammers be fixed at P .

- Derive an expression for the secrecy capacity of the transmitter assuming each friendly jammer transmits with equal power.
- What is the optimal power allocation to each friendly jammer in order to maximise secrecy capacity?

- s** 10.13 Consider the plot of the SNR at the legitimate receiver (bold line) and the eavesdropper (dotted line) as depicted in Fig. 10.23. Mark the time-slots for opportunistic secure communications.

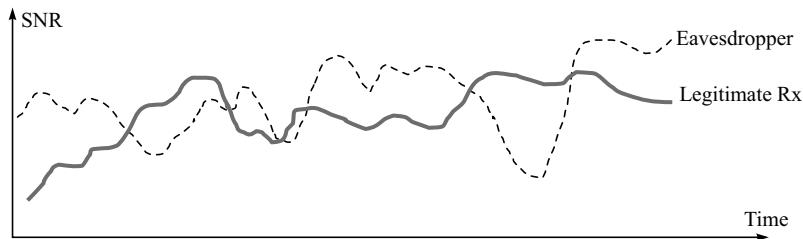
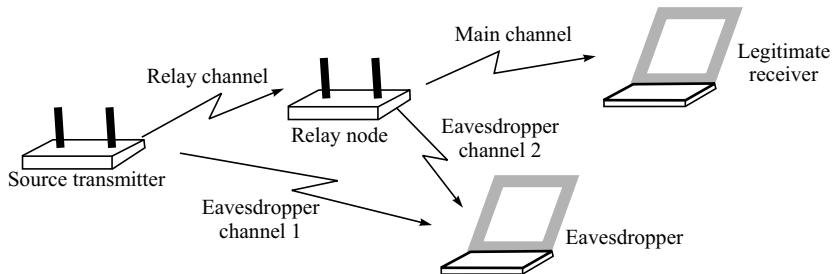


Fig. 10.23

- s** 10.14 Consider the relay-based communication scenario, shown in Fig. 10.24 (a). The SNR at the legitimate receiver (bold line), the eavesdropper channel 1 (dotted line) and eavesdropper channel 2 (dash-dot line) are plotted in Fig. 10.24 (b). Mark the time-slots for opportunistic secure communications.

(a)



(b)

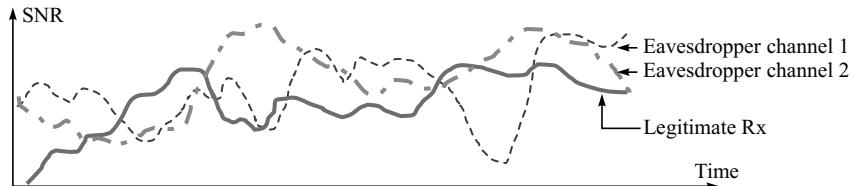


Fig. 10.24

- M** 10.15 Consider a wireless communication scenario where there exists a flat Rayleigh fading channel between the transmitting source and the destination (with channel gain h_{sd}), as well as, the source and the eavesdropper (with channel gain h_{se}). Find the probability of intercept, i.e., $P(C_s < 0)$.
- D** 10.16 Consider a friendly jammer shown in Fig. 10.25. The jammer transmits a random codeword from a Gaussian codebook with average power, P_J , and this transmission is received both by the eavesdropper and the legitimate receiver. Derive an expression for the secrecy capacity of the transmitter.

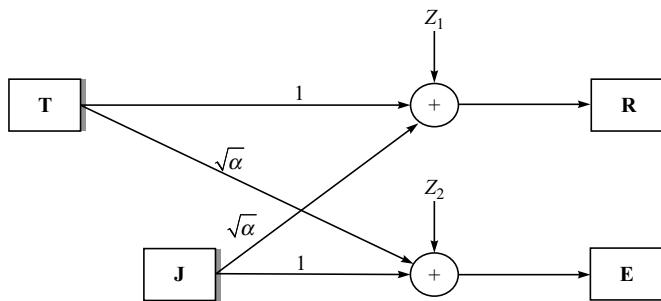


Fig. 10.25

COMPUTER PROBLEMS



- 10.1 Consider a simple wireless communication scenario where there is no fading. Let the noise in the main channel and the eavesdropper channel be complex AWGN with variances σ_m^2 and σ_e^2 respectively. Generate a mesh curve for the secrecy capacity, using the two variables σ_m^2 and σ_e^2 .
- 10.2 Consider a wireless communication scenario where there exists a flat fading channel between the transmitter and intentional receivers, as well as, the transmitter and the eavesdropper. Assume that both the channels can be modelled as Rayleigh fading, with AWGN with variances σ_m^2 and σ_e^2 respectively. Generate the outage probability curves (similar to those in Example 10.7) for the following combinations.
- $\text{SNR}_m = \text{SNR}_e = 30 \text{ dB}$.
 - $\text{SNR}_m = \text{SNR}_e = 10 \text{ dB}$.
 - $\text{SNR}_m = 30 \text{ dB}$ and $\text{SNR}_e = 20 \text{ dB}$.
 - $\text{SNR}_m = 20 \text{ dB}$ and $\text{SNR}_e = 10 \text{ dB}$.

Compare and comment on your results. If the difference in the SNR of the main and the eavesdropper channel is the same (say 10 dB), which is a more conducive scenario: Low SNR or high SNR from the perspective of secrecy capacity?

- 10.3 Consider a wireless communication scenario as shown in Fig. 10.25 with a transmitter, T, a legitimate receiver, R, an eavesdropper, E, and a friendly jammer, J. Generate a 2D mesh curve for the secrecy capacity as a function of the transmitter power, P_T and the jammer power, P_J . Assume that there is a limit on power, i.e., $P_T + P_J = P$. Write all the assumptions that you make.
- 10.4 Consider a wireless communication scenario as shown in Fig. 10.25 with a transmitter, T, a legitimate receiver, R, an eavesdropper, E, and a friendly jammer, J. We wish to explore the effect of the path loss exponent. Model the received power $P_R = \frac{KP_T}{d^n}$ where P_T is the transmit power, d is the distance between the transmitter and receiver, n is the path loss exponent and K is a constant. Generate 2D secrecy capacity curves for different values of n , as a function of the transmitter power, P_T and the jammer power, P_J . Assume that there is a limit on power, i.e., $P_T + P_J = P$. What is the relation between the secrecy capacity and n ?
- 10.5 Consider a wireless scenario with multiple antennas at the transmitter. Suppose the transmitter sends a sum of two signals: information signal and artificial noise. Let us assume that the channel is Rayleigh fading. Let α (< 1) be the fraction of the total power transmitted as artificial noise. Plot outage probability curves for the following combinations:
- (i) $\text{SNR}_m = \text{SNR}_e = 30$ dB.
 - (ii) $\text{SNR}_m = \text{SNR}_e = 10$ dB.
 - (iii) $\text{SNR}_m = 30$ dB and $\text{SNR}_e = 20$ dB.
 - (iv) $\text{SNR}_m = 20$ dB and $\text{SNR}_e = 10$ dB.
- Compare and comment on your results. What assumptions have you made?

PROJECT IDEAS

- 10.1 **Spatial Secrecy Map:** Write a program that can take as input the locations of the transmitter, T, a legitimate receiver, R, an eavesdropper, E, and a friendly jammer, J. The relative locations can be altered on the 2D grid. If required, the transmitter can send the sum of two signals: information signal and artificial noise. The program should output a spatial secrecy map (either colour-coded or grey scale) that depicts the level of secrecy capacity at different locations.
- 10.2 **Relays and Secrecy Capacity:** Consider the system model shown in Fig. 10.26, where there are M relays between the transmitter, T and the destination, D. Assume that there is a constraint on the total power (P_T), to be distributed between the transmitter, T, and the relays R_i , $i = 1, 2, \dots, M$. Assume half-duplex relaying such that, in the first half of the transmission time, the source transmits while the relays transmit in the second half. Also, assume that the channel state information (CSI) of the eavesdropper as well as the CSI of the destination are available at the relays. Each relay amplifies and forwards the signal that it receives. Find the secrecy rate for transmitter, T. What is the best way to choose the relay amplification (weights) in order to maximise the secrecy rate? Write a computer program to mimic the scenario. Make the required assumptions. Does it make sense to do relay selection?

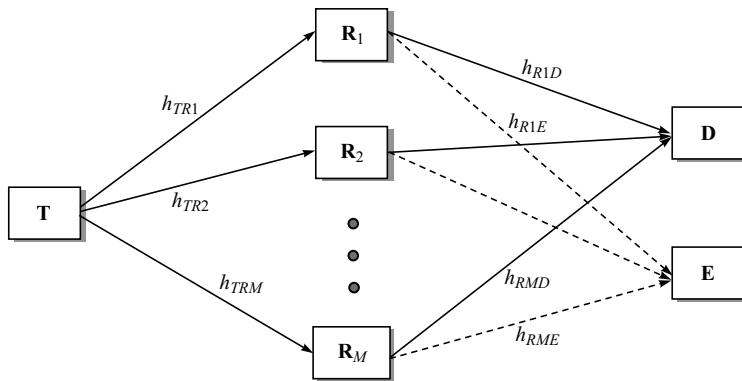


Fig. 10.26

10.3 Multiple Access Channel: Consider N users communicating with an intended receiver, **R**, in the presence of an eavesdropper, **E**, as shown in Fig. 10.27. Find the secrecy capacity of the multiple access channel.

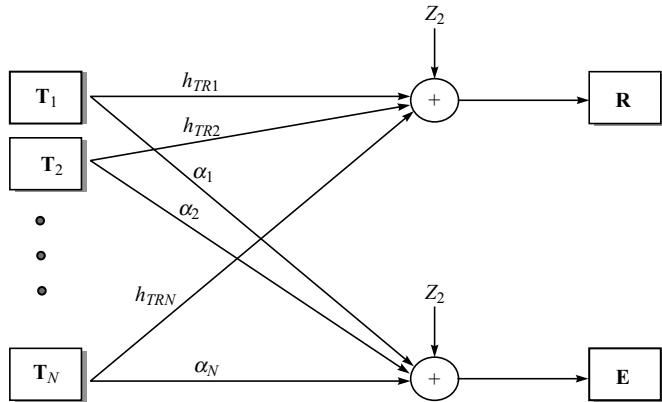


Fig. 10.27

10.4 Secrecy in AF Relay System: Determine the secrecy outage probability of a dual-hop amplify-and-forward (AF) relay system when the single cooperating relay among the relays is selected without the knowledge of eavesdropper's instantaneous channel state information (CSI). Assume Rayleigh fading.

REFERENCES FOR FURTHER READING

1. Bloch, M. and Barros, J., *Physical-Layer Security*, Cambridge University Press, 2011.
2. Liu, R. and Trappe, W., *Securing Wireless Communications at the Physical Layer*, Springer, 2010.

Index

Symbols

3-D cipher 418
3GPP-LTE 310, 316

A

Abelian group 125
AC coefficients 56, 57
Acceptable bit error rates 82
Achievable rate 99, 112
Additive White Gaussian Noise (AWGN) Channels 342
Achievable 84, 85, 87, 99, 107, 112, 428, 429, 432, 444, 446
ADFGVX
 cipher 416, 417
 fractionation 410, 416
ADFGVX fractionation cipher. 410
Advanced Encryption Standard (AES) 382, 385, 412
Affine cipher 377, 378
Alamouti code 244, 246, 247, 250, 251, 257, 262, 267, 269, 270, 273, 274
Alamouti scheme 156, 158, 160, 162, 163
Algebraic construction of LDPC codes 150
Algebraic attack 407
Algorithms 375, 379, 387, 420
Amplify-and-Forward (AF) 76, 109
ANSI
 standard X9.32. ANSI X9.9 382
 X9.23 382
 X9.31 393
 X9.42 402

Arithmetic

 code 36, 55
 encoding 156, 164
 coding 40, 63

Artificial Noise Forwarding 440

Asymmetric Cryptographic Techniques 387

Asymmetric
 encryption 374
 (Public-Key) Algorithms 387
Asymptotic Coding Gain 143, 332, 358
Attacker 372
Augmented Generating Function 292, 348
Authentication 373, 374
Automatic Link Establishment 188, 192
Automatic Link Establishment (ALE) protocol 188
Automatic Repeat Request 143
Avalanche effect 378
Average
 Conditional Entropy 23
 Conditional Self Information 20
 Mutual Information 18, 22
 Number of Nearest Neighbours 343
 Self Information 18

AWGN Channel 100, 105, 106, 111, 155, 168,

B

Bahl, Cocke, Jelinek and Raviv (BCJR) Algorithm 310,
 311
Bandwidth
 efficiency diagram 95, 96, 114
 limited channels 327

- BB84 297
 Base field 208, 209, 212, 213, 236
 Berlekamp-Massey-Forney 222
 Bigram 19
 Binary entropy function 19, 24
 Binary
 erasure channel 74, 80
 erasure wiretap channel 425, 426, 447
 tree 19
 Binary Golay Code 188
 Binary input Q-ary output Discrete Memoryless Channel 73
 Binary symmetric channel (BSC) 15, 22, 74
 Biometric Encryption 372, 404, 405, 413
 Biometrics 404
 Birthday paradox 396
 Bit flipping algorithm 151, 152
 Bits 12, 14, 15–17, 19, 20, 23, 26–28, 30–44, 46–51, 54, 55, 57, 59, 63, 65, 68–70
 Block
 Ciphers 380
 code 122–124, 126, 127, 131, 132, 135, 136, 141, 153, 156, 160, 162, 166, 167
 codes 83, 86
 Fading Model 436
 length 122, 124, 131, 135, 141, 147, 149, 152, 154, 160, 161
 Blocklength 280, 285
 BMP 44, 63
 Bose-Chaudhuri Hocquenghem (BCH) 207, 208
 Bounds on Minimum Distance 153
 Branch metric 298, 342
 Broadcast Channel 74, 77
 Brute force attack 405, 413
 Burst error correction 185, 198, 204, 205
 BSC 15, 17, 21, 22
 Burst errors 170, 185–187, 191, 192, 200, 204
- C**
- Caesar cipher 376, 377
 Capacity 72, 74, 78, 87, 89, 90, 96, 97, 98, 99, 108, 110, 114
 Boundary 96
 for MIMO Systems 97, 443
 of the Channel 78
 of a DWTC 429, 430, 444
 Catastrophic
 Convolutional Code 286
 Error Propagation 289
 CCITT 56
- Complete Decoder 135, 300
 Chain rule 21, 23, 52, 62
 Channel
 Capacity 78
 Capacity for MIMO Systems 97
 Capacity Theorem 90, 108
 Coding 72, 74, 81–84, 108, 120, 134, 143, 159
 Coding Theorem 72, 74, 82, 84, 108
 Decoder 83
 Encoder 83
 state information 349, 350, 352, 354, 355, 359
 State Information (CSI) 93, 97
 Transition Matrix 75
 Transition Probability 75
 Transition Probability Matrix 75
 Chaos functions 402, 413
 Check nodes 151, 152
 Chien search 222
 Chosen-ciphertext 406, 413
 Chosen-plaintext 406, 413
 Chrominance 57, 58
 Cipher 305, 371, 372, 373
 Ciphertext 373, 376, 386, 406, 413, 417
 Ciphertext-only 406, 413
 Circuit
 energy 231
 Implementation 168
 Implementation of Cyclic Codes 193
 Claude E. Shannon 12
 Code 12, 20, 26–46, 55, 56, 61, 63–70, 121–124, 126–156, 158–168
 Codebook 83
 Coded Modulation 275, 296
 Codes 327, 328, 330, 338, 342, 343
 Codeword Frames 278, 289, 317
 Code Division Multiple Access (CDMA) 96
 Coded Modulation Technique 328
 Code excited linear prediction (CELP) 20
 Code rate 83, 86, 123, 128, 132, 144, 154, 160, 162, 163, 166
 Codeword 83, 90, 101, 121–132, 134–141, 145, 146, 149, 151, 153, 155, 156, 160–162, 165, 167, 168
 Coding gain 143, 156, 243, 244, 249, 267, 269, 270, 332, 358
 Coding Gain Distance 249
 Coding strategy 245
 Coefficient matrix 130
 Collision 395, 397
 Complete Decoder 135, 300

- Combining scheme 246
Complexity
 Class co-*NP* 391
 Class *NP* 391
 Class *P* 391
 Complex Orthogonal Design 158, 256, 257, 264
 Components 122, 126, 128, 129, 131, 149, 151, 161
 Compression 11, 12, 26, 30, 42, 54, 55
 Compression permutation 383
 Computational energy 231
 Computation of d_{free} 348
 Concatenated Codes 168
 Concatenation 308, 313
Conditional
 Entropy 20, 23
 probabilities 74, 75, 107
 Self-Information 17
 Confidentiality 373, 397
 Confusion 375, 411, 415, 425
 Congruent modulo 172
 Conjugacy class 214
 Conjugates 213–215, 236
 Constraint Length 279, 285
 Continuous random variables 22, 62
 Convex 24, 48, 53, 63, 65
 Convolutional Code 280, 286
 Cooperative Jamming 439
 Coset 137–142, 161
 Coset leader 138–142
 Cracking 372
 Critical Rate 84, 108
 Cross Interleave Reed-Solomon Code (CIRC) 233, 234
 Crossover probability 79, 87, 99, 106, 109, 112, 115
CRYPT 382
 Cryptanalysis 371, 372, 405, 412, 414, 419
 Cryptanalyst 372
 Cryptography 371
 Cryptosystem 371
 Cumulative Distribution Function 39
 Cutoff Rate 74, 103, 110
 Cyclic 169–171, 175–189, 193, 195, 198–205
 Cyclic burst 185, 187, 200
 Cyclic code 170, 176–186, 188, 189, 195, 199, 200–205
 Cyclic Redundancy Check (CRC) Codes 189
- D**
- D-adic 36, 40
 Data Encryption Standard (DES) 371, 372, 379, 381, 382, 412
 DC coefficient 56, 57
 DCT 56–60, 65, 66, 69
 Decode-and-Forward (DF) 76, 109
Decoding
 BCH codes 117, 198, 207, 209, 214
 failure 300
 of a linear block code 131, 134, 167
 of BCH codes 171, 208, 214, 220, 224, 226, 235
 Spheres 90, 135
 strategy 83, 86
 Window Width 289
 Decryption 371, 373, 384
 Degraded Wiretap Channel (DWTC) 425, 443
 Degree 170–175, 177, 178, 180–182, 185–191, 199–204
 Deinterleaver 308
 Delay Optimal. 254
DES
 Decryption 384
 Encryption 383
Design
 criteria 248, 327, 328, 353, 356
 rules for TCMs for fading channels 352, 359
 Designed distance 216–219, 227, 236, 237, 239
 Design rules for TCMs 352, 359
DESX 384
Determinant
 criteria 243, 244, 249
 criterion 356, 359, 361
 Diagonally Orthogonal Space-Time Block Code 259
Differential
 cryptanalysis 406
 Entropy 23
 Diffie-Hellman Key Agreement Protocol 400
 Diffusion 375, 411, 415
Digital
 Signature Algorithm (DSA) 374
 Signatures 374
 Signature Standard (DSS) 374
 Dimensionality 101, 105, 106
Discrete
 Cosine Transform 56
 logarithm Problem 400, 401, 413
 input, Discrete-output channel 74, 75, 110
 Memoryless Channel (DMC) 75
 Distance Bounds for Convolutional Codes 301
Distance
 Bounds 12, 36, 153, 154, 160, 277, 278, 289, 301–303, 316, 343
 Criterion 356
 Notions for Convolutional Codes 289
 Distortion 47–49, 63

- Rate Function 49
 Distributed computing 392, 406
 Diversity
 gain 243, 244, 249–251, 259, 264, 266, 267, 270
 Order 156
 Division Algorithm for Polynomials 172
 DSC-HDTV 226
 Dual code 183, 202
 Dual of the code 130
- E**
- Eavesdroppers 372, 373, 403
 Eavesdropper's channel 423, 425, 427, 430, 433–435, 438, 439, 443–446, 449
 ECMA standard 240
 E-commerce 407
 Effective Length 351
 Efficiency 30, 70, 71
 Elliptic Curve Cryptography (ECC) 398, 413
 Elliptic Curve Discrete Logarithm problem 400, 413
 Encoder for STTC. 355
 Encryption 371–375, 379, 381–383, 385, 404, 405, 407, 410, 412, 413, 420
 Energy consumption 231
 Enigma machine 410
 English language 19
 Enigma ciphers 410
 Entropy 12, 18–26, 29–40, 47, 49, 51–56, 59, 60, 62–67, 70
 Entropy function 79, 109
 Entropy
 Rate 50, 52
 Rate of a Stochastic Process 50, 52, 64
 Equivalent 128, 129, 132, 133, 161, 164, 282, 286, 287, 288, 317, 319
 Equivalent Codes 128
 Equivocation 425, 427, 428, 429, 444
 Equivocation Rate 427
 Erasures 136, 137, 144, 166
 Error
 coefficient 349
 control code 83, 120, 144, 169, 231
 Control Coding 83
 correcting codes 119
 event 329, 333, 349
 Event Probability 343
 locator polynomial 221, 222, 236
 Propagation 289
 state diagram 344–348
 vector 134, 137, 139, 140, 141
- Weight Matrix 344
 Euclidean distance 329–336, 339, 340, 342–345, 348, 349, 352, 356, 358, 360, 361–363
 Examples of BCH codes 216
 Expansion permutation 383
 Euclidean division algorithm 172
 European Computer Manufacturers Association (ECMA) standard 192
 Exponential key agreement 400, 413
 Extended Hamming Code 168
 Extension field 208–216, 218, 226, 236, 241
 Extrinsic information 312, 313
- F**
- Fades 349
 Fast Rayleigh Fading 356
 FAX images 44
 Federal Information Processing Standard (FIPS) 46–3 382
 Feistel cipher 380, 382, 385
 Fermat's Little Theorem 390
 Field 125–127, 133
 Finite
 Impulse Response (FIR) 194
 State Machine 390
 FIPS 46–1, 46–2, 46–3 382
 Fire code 187, 195, 200, 204, 205
 First event error probability 303
 Fixed Length Code (FLC) 26
 Flag byte 45
 Fountain Codes 241
 Frame check sequence (FCS) 189, 200
 Free
 distance 313, 315, 318, 320, 323, 325, 328, 330, 331
 Euclidean Distance 330
 length 290, 291, 318
 Space Optical (FSO) Communications 274
 Frequency Flat Fading 97
 Friendly jammer 421, 422, 439, 340, 442, 445, 447, 449–451, 452
 Full Rate 250, 251, 254–256, 260, 263, 267, 271
 Full diversity 249
 Full Rate Orthogonal Design 254
 Full Secrecy Rate 429, 444
- G**
- Gallager code 149, 161
 Galois Field 125
 Gaussian Channel 88, 107, 110
 Gaussian Wiretap Model 432
 Generalised Complex Orthogonal Design 158, 257, 264

Generalised Real Orthogonal Design 157, 253, 254, 264
 Generating cyclic codes 177
 Generating function 278, 291–293, 318, 322
 Generator
 matrix 127–133, 138, 144, 148, 149, 159, 161, 164–168, 252
 matrix for the convolutional code 293, 318
 polynomial 177–180, 182–187, 190, 195, 196, 198–205, 207, 211, 215–220, 223, 225–229, 235–241, 284
 Polynomial Matrix 284, 288
 GIF 45
 Gilbert and Varshamov Bound 154
 Global System for Mobile (GSM) 380
 Golay Codes 188
 Gorenstein–Zierler decoding algorithm 223
 Gray coding 247, 362
 Gorenstein-Zierler 220, 223
 Graphics Interchange Format (GIF) 44
 Ground field 208, 236
 Guaranteed secrecy capacity 441

H

Hackers 372, 379
 H.261 58, 241
 H.263 58, 65
 H.264 58, 71
 Hacking 372
 Hamming
 Bound 146–149, 155, 164, 166–168
 Codes 133, 147–149, 155, 160, 161, 163, 164, 166–168
 Distance 121–123, 134, 137, 160–164
 Distortion 47
 weight 121, 122, 131, 150, 160, 162, 163
 Hard decision decoding 73, 110
 Hash 375, 395, 396, 408
 value 343, 348, 349
 HDTV 58, 226
 Heisenberg Uncertainty Principle 403, 413
 Heller 301–303, 305, 306, 316, 318, 322–325
 Heller bound 302, 303, 322–325
 Hermitian 97
 Homophonic substitution 376
 High-Level Data Link Control (HDLC) 192
 Hubble space telescope 225
 Huffman
 code 35, 36, 40, 41, 43, 44, 46, 55, 61, 63, 65–70
 coding 9, 11, 12, 20, 26, 27, 29, 31, 32, 33,

encoding 1, 12, 20, 31, 36, 37, 38, 39, 40, 41
 coding algorithm 31, 59
 tree 32
Hybrid
 concatenated convolutional codes (HCCC) 308
 cryptosystems 372, 392, 398, 404, 407, 411, 413
 FEC/ARQ 306
 RS-ARQ Scheme 241

I

IEEE 802.3 Standard 149
 IEEE 802.11ac 266
 Image Compression 11, 12, 44, 45, 54, 55, 64, 69
 IEEE 802.16 205
 IEEE 802.16e 152, 158
 IEEE 802.16e WiMax 152
 Improved PES (IPES). 385
 Incomplete decoder 135
 Indeterminate 283
 Information
 capacity theorem 72, 74, 87, 89, 90, 108, 110
 frames 57, 59, 65, 100, 278, 279, 289, 300
 Radius 24
 Rate Distortion Function 48
 Theory 9, 11, 12, 61, 71
 word 127, 132
 Instantaneous Codes 27, 28
 Interleaved 349
 Interleaver 278, 308, 315
 design for turbo codes 277, 278, 308, 315
 Interlopers 372
 International Data Encryption Algorithm (IDEA) 371, 372, 379, 381, 385, 412
 International Standard Book Number 135
 Inter-symbol interference 349
 Irreducible 173–175, 178, 199, 203, 212, 213, 215, 236, 238, 241
 Irregular LDPC code 150
 ISO 56
 ISO 11166 393, 402
 Iterative
 Decoding Suitability (IDS) 315
 MAP decoding 310, 311
 ITU – Optical Transport Network 123
 ITU-T video compression standard 58
 Iwadare Code 326

J

Jensen Shannon Distance 24
 Jensen Shannon divergence 24
 Jensen's inequality 24

- Jigsaw transform 418
 Joint Entropy 20
 JPEG image compression 55
 Standard 304, 306, 313, 340, 341, 361, 363, 365, 371, 372, 374, 379, 381, 382, 385, 388, 393, 396, 398, 402
 JPG 44
- K**
- Kerberos 382
 Kerckhoff's Principle 375, 412, 414
 Key 371–389, 392–398, 400–408, 410–420
 Key Management 379
 Key patents in the area of cryptography 407, 408
 Knapsack cipher 417
 Known Good Convolutional Codes 304
 Known-plaintext 406, 413
 Kraft Inequality 28, 64
 Kullback Leibler (KL) Distance 23
- L**
- Large prime numbers 388, 389, 390, 413, 415
 Leakage of Information 423, 425
 Lempel-Ziv Algorithm 42
 Lempel–Ziv encoding 11, 12, 31
 Linear
 code 123, 124, 126–128, 130, 132, 149, 151–153, 155, 160, 161, 164–166
 block 14, 36, 37, 38, 57, 67, 70, 83, 86, 94, 95, 101, 105, 117, 119, 120, 121, 127, 130–132, 134, 135
 block Codes 89, 96, 117, 119, 121, 126, 130, 130, 132, 134, 145, 155, 156, 168, 169, 170, 207, 235, 243, 244
 cryptanalysis 407
 processing 156, 158
 Lloyd–Max quantiser 41
 Long Term Evolution 98, 115
 Long Term Evolution (LTE) Advanced 98
 Lossless compression 54, 55, 56, 57
 Lossy compression 54, 56, 58
 Low Density Parity Check (LDPC) 121, 149, 160, 161
 Low Density Parity Check (LDPC) Codes 121, 149, 160
 Lucifer cipher 408
 LTE – Advanced 158
 LUCIFER project 410
- M**
- Main channel 422, 424, 425, 430, 431, 433–435, 436, 439, 443, 447, 449, 451
 Majority Decoding 86
 Majority Logic Decoding 295
 Mangalyan X 155
 Man-in-the-middle Attack 402
 Mapping by set partitioning 333
 Markov
 Chain 51, 52
 Markov Process 51
 Mason's gain formula 292
 Mars mission 225
 Mars Reconnaissance Orbiter (MRO) 316
 Mathematical measure of information 13
 Matrix
 description of convolutional Codes 293
 description of cyclic codes 180
 description of linear block codes 293
 Maximum
 a posteriori (MAP) algorithm 310, 317, 318
 distance code 132, 161, 163
 Distance Separable (MDS) code 227, 237, 238, 153
 diversity 352, 356–358, 359, 367, 368
 likelihood (ML) decoding 156
 likelihood criterion 342
 Maximum Likelihood 156
 MD4 396
 MD5 393, 396, 413
 Meggitt Decoder 197, 205
 Memory 113, 139, 169, 193, 195, 196, 278
 MELP/CELP 20
 Message
 Authentication Code (MAC) 379
 Digest 375, 396
 integrity 279, 372, 373, 379, 397
 Minimal Polynomial 212
 Minimum distance 123, 124, 127, 131, 132, 134–137, 141, 145, 146, 148–151, 153, 154, 160, 161, 164–167
 Minimum
 distance 289
 key length 381, 383, 385, 386
 weight 57, 121–125, 131, 132, 134, 136, 138, 139, 141, 142
 distance of a code 123, 132, 160
 key requirements 381
 weight of a code 123, 131, 160
 Modified Bahl, Cocke, Jelinek and Raviv (BCJR) Algorithm 311
 Modified BCJR 311
 Modulator 73, 74, 75, 101, 231, 328, 329
 Monic 171, 173–178, 182, 196, 199, 201, 203

- More capable channel 422
More capable 422, 434
Moving Picture Coding Exports Group 58
MPEG-1 58
Multi Phase Shift Keying 327
Multiple
 access channel 74, 76, 77, 98
 input multiple output (MIMO) 76, 110
Multiple Input Single Output (MISO) 76
Multiplicative cipher 377
Mutual Information 15, 18, 22
- N**
- NASA 188, 192
National Security Agency (NSA) 379, 384
Nats 14
Nearest Neighbour decoding 134
Nest 233
Nested Codes 233
Next distance 349
N-grams 19
No free lunch 46
Noisier 422, 434
Noisy Channel Coding Theorem 72, 74, 82, 84
Non-catastrophic Convolutional Code 286
Non-linear nature of TCM 331
Non-repudiation 373
NSB 382
Nonce 375, 415
Number Field Sieve 392
- O**
- One Time Pads 397, 423
One-Way Hash Function 395
One-way Hashing 297
Optimal linear codes 152
Optimal code 152, 153, 161, 163
Optimum
 quantiser design 49
 quantizers 49
Order 207, 209–211, 215, 216, 227, 229, 236, 238
Ordinary Error Propagation 289
Orthogonal Design 157
Outage 437
 Capacity 437
 Probability 437
- P**
- Pairwise error probability 244, 248, 267
Parallel
 Concatenated Convolutional Codes (PCCC) 308
 Gaussian channels 74, 92, 93
 transitions 334–336, 345
Parity check matrix 130, 131, 140, 144, 147–151, 153, 155, 161, 163–166
Parity
 Check Matrix 286
 Check Polynomial 183
 Matrix 129
Pass-phrase 295
Password 295
Parity symbols 131, 132, 163
Path in the trellis 283, 292
Path metric 298, 299
Pattern attack 406
PCX Format 36
PCX 44, 45, 63
Perfect Codes 141, 145, 147, 155, 161, 164, 166, 167
Perfectly secure 422, 423, 443, 446
Perfect Secrecy 429
Performance
 bounds 216
 of Alamouti code 250, 251
 of RS codes over real channels 179
Permutation 128, 129
Persymmetry 222
Phrases 42, 43
PKCS #7 299
Plaintext 373, 376, 380, 412, 417
Playfair cipher 378, 379
Plotkin Bound 154
PNG 44, 45
Polar Codes 168
Politics of Cryptography 407
Polybius checkerboard 409
Polynomial 170–172
 codes 190
 description of convolutional codes 283
 time 391
Power limited channels 327
Portable Network Graphic (PNG) 44
Prefix Code 28, 40
Prefix Condition 27–31, 63, 64
Pretty Good Privacy (PGP) 371, 372, 387, 388, 393, 413
Primality Decision Problem 390
Prime
 counting function 389
 density theorem 390
 factors 388
 numbers 388

Prime polynomial 173–175, 187, 194, 199–201
 Primitive BCH codes 214
 Primitive Blocklength 211
 Primitive Cyclic Code 211
 Primitive Element 208
 Primitive Polynomial 210
 Private key 374, 387–389, 392, 393, 395, 396, 398, 401, 408, 413
 Probability of error 139, 141–144, 164, 167
 Probability Transition Matrix 75
 Product Criterion 356
 Product-distance Criteria 356
 Pseudo Markov Huffman (PMH) 69
 Pseudo-random number generator 393
 Proposed Encryption Standard (PES) 385
 Pseudo-random generation algorithm (PRGA) 386
 Public key 374, 387–389, 392, 393, 395, 396, 398, 408, 413, 415, 417, 418
 Public-key Algorithms 387
 Public Key Cryptography 374
 Public-Key Cryptography Standards (PKCS) 388
 Pulse amplitude modulation 136
 Puncturing 184

Q

QCIF 58
 Quadrature Amplitude Modulation 327
 Quantization 47, 59, 64, 65
 Quantum Cryptography 372, 403, 413
 Quantum entanglement 403
 Quasi
 cyclic code 184, 200
 orthogonal space-time block codes
 (QOSTBCs) 260
 static 250, 251, 263

R

Rail fence cipher 416
 Random
 construction of LDPC codes 150
 interleaver 315
 Random Selection of Codes 100
 Rank and Determinant Criteria 249
 Rank criteria 249
 Rank criterion 356, 359
 Rank-determinant Criteria 356
 Rate Distortion Function 47, 48
 Rate-Equivocation Region 428
 Rate of a space-time block code 249

Rayleigh distribution 350
 RC2 386, 407
 RC4 386, 387, 407, 412, 415
 RC5 386
 RC Ciphers 371, 379, 386, 387
 Real Orthogonal Design 157, 252–254, 264
 Reciprocal 205
 Reducible 173, 174, 199
 Redundancy 120, 121, 123, 153, 156, 161
 in spoken English 20
 of the English language 19
 Reed-Solomon (RS) codes 207, 225, 228
 Reed-Solomon encoders and decoders 228
 Regular LDPC code 150, 151
 Relative Entropy 23, 62
 Relay Channels 76, 109
 Reliability condition 428
 Repetition code 85, 86, 114, 115
 Residual Error Rate 141
 Residue 172, 199, 204
 Restricted Algorithm 375
 Rician fading 350, 363, 364, 368
 Rician Parameter 350, 367
 Rieger Bound 186
 Rijndael cipher 385, 412
 Ring 125, 126
 Ring of polynomials 172
 Rotor machine 410
 Rivest Cipher 386
 Ron's Code 386
 Rotor cipher machines 410
 RSA Algorithm 388
 Run 44–46, 59, 60, 63, 64, 68, 69
 Run-length Encoding 44, 63
 Running cipher 282

S

S-boxes 383, 384, 387, 415
 Scalar
 channel capacity 97
 transfer function 345
 Scytale 409
 Secrecy Capacity 429, 434, 452
 Secrecy Capacity of a DWTC 429
 Secret Key Algorithms 379
 Secret Key Cryptography 374
 Secure Hash Algorithm (SHA) 396
 Secure Mail 397
 Secure Multipurpose Internet Mail Extensions (S/MIME) 397

- Security
 of DES 384
 of PGP 395
 of RSA 392
 provided by IDEA 386
 Self-dual code 130, 149
 Self Information 13, 18, 20, 37
 Sequential Decoding 295
 Serial Concatenated Convolutional Codes (SCCC) 308
 Set
 of conjugates 214
 partitioning 333, 358
 partitioning tree 336
 Shannon Codes 39
 Shannon-Fano-Elias encoding 11, 12, 31, 39
 Shannon
 Code 39
 Limit 95, 96, 110
 Fano–Elias 4
 Shannon’s Notion of Security 422
 Shortened cyclic code 184, 200, 201
 Shortening 256, 268, 270
 Signal energy (radio energy) 231
 Simple decoding 250, 267, 269
 Single Input Multiple Output (SIMO) 76
 Single Input Single Output (SISO) 75
 Single Key Algorithms 379
 Single Key Cryptography 374
 Singleton bound 132, 153
 Sliding Block Code 280
 Slow Rayleigh Fading 355
 Soft
 decision decoding 73, 110
 output viterbi algorithm (SOVA) 308
 Source
 Coding 11
 Coding Theorem: 29
 Space
 Time Block Code 244
 Time Block Codes 155
 Time Code Design Criteria 248
 time encoder. 244
 Time Trellis Codes (STTC) 327, 328, 353, 359
 Spatial Multiplexing Rate 249
 Spatial Secrecy Map 452
 Sphere Packing Bound 146
 Sphere Packing Problem 90
 Squared
 distortion 47
 error distortion 47
 Squared Product Distance 351
 Square root bound 189
 S-Random Interleaver 315
 STANAG 5066 143
 Standard array 138–142, 164
 Stationary 47, 48, 51, 53, 64
 Stationary Markov Chain 52
 Statistical attack 406
 STBC design targets 263
 Steganography 397
 Steps for decoding BCH codes 222, 236
 Stream cipher A5/1 380
 Stream Ciphers 380
 Strictly convex 24
 Strong Secrecy Capacity 429
 Subfield 208, 209, 211, 212, 236, 239
 Substitution 376, 383, 419
 Super-Orthogonal Space Time Trellis Codes 368
 Super Polynomial Time 391
 Survivors 299
 Symbol error rate 156
 Symmetric 74, 80, 81
 Symmetric Algorithms 379
 Symmetric Channel 429, 430
 Symmetric Cryptography 374
 Synchronous Transmission Standards 192
 Syndrome 121, 139, 140, 143, 151, 152, 161, 163, 164
 Syndrome
 Decoding 140
 Polynomial 179
 Polynomial Vector 286
 Syndromes 140
 Systematic code 131, 132, 161, 162, 164
 Systematic Convolutional Encoder 284
 Systematic Encoder 286
 Systematic Form 129
- T**
- Tanner Graph 151
 TCM Decoder 342
 TCM for Fading Channels 349
 Telephone channel 89, 100
 The division algorithm for polynomials 172
 The square root bound 189
 Third Generation (3G) 96
 Ternary Golay Code 189
 Third Generation Partnership Project (3GPP) 315
 Threshold Decoding 295
 Time diversity of the TCM scheme 352
 TIFF, 44, 63

- Time Varying Binary Channels 115
 Toeplitz matrix 181, 183, 202
 Transfer function 284, 345
 Transition Probability Matrix 75
 Transmission matrix 254, 257–261, 268
 Transposition 377, 416
 Tree Code 280
 Trellis
 Coded Modulation (TCM) 327–329
 Coded Spatial Modulation 367
 Codes 278, 282
 Diagram 282
 Trivial perfect code 147
 Turbo Codes 308, 315
 Turbo Decoding 310
 Turbo Product Code 313
 Turing Machine 390
 Type-I hybrid FEC/ARQ 306
 type-II hybrid FEC/ARQ 306
 TYPEX 410
- U**
 Ultra Wideband (UWB) communication 240
 Ultra Wideband (UWB) 240
 Ungerboeck 335, 336, 339, 342, 343, 347, 356, 358, 360, 361, 363, 365
 Ungerboeck's TCM Design Rules 336
 Uniquely Decable Codes 27
 Universal source coding 42
 UNIX operating system 43, 61
 Unpredictability 402
 U.S. MIL-STD 20
- V**
 V.17 365
 V.32 341
 V.34 340, 341
 V.34 + 341
 V.34 modem 340
 V.90 341
 V.92 341
 Vandermonde matrix 239
 Variable Length Code (VLC) 26
- Vector Viterbi Algorithm 354
 Vernam's Cipher 423
 Video Coding Experts Group 58
 Video Compression Standards 57
 Vigenère cipher 376, 409, 417
 Vigenère square 376
 Viterbi
 algorithm 328
 decoding 277–289, 295–300, 316, 318, 320, 329, 360, 364, 366
 Voyager II 225, 235
- W**
 Water
 Filling 94, 110
 Pouring Algorithm 94, 98, 108
 Waveform Channels 75
 Weakly Symmetric 74, 81, 429
 Weak
 Rate-Equivocation Pair 428
 Rate-Equivocation Region 428
 Secrecy 424
 secrecy condition 428, 431
 secrecy constraint 424, 432, 444, 446
 Weight Distribution 168
 Wilson Leung Code 365
 Wiretap model 421–424, 432
 Word 121, 124, 126, 127, 130, 132, 134, 135, 137, 139, 140–143, 145, 151, 160, 164, 165, 168
 Word length 280, 285
 Wyner-Ash code 323
 Wyner's wiretap model. 424
- X**
 X9.30–1 396
 X9.30–2 396
 X9.52 382
 XOR 373, 377, 383, 385–387
- Z**
 Zero mean gaussian noise process 349
 Zeros 105, 215
 Zig-zag coding 57, 58, 60, 69