

Monica Borda

Fundamentals in Information Theory and Coding

 Springer

Fundamentals in Information Theory and Coding

Monica Borda

Fundamentals in Information Theory and Coding

Author

Prof. Dr. Eng. Monica Borda
Technical University of Cluj-Napoca
Dept. Communications
Str. Baritiu 26-28
400027 Cluj Napoca
Romania
Telephone: 0040-264401575
E-mail: Monica.Borda@com.utcluj.ro

ISBN 978-3-642-20346-6

e-ISBN 978-3-642-20347-3

DOI 10.1007/978-3-642-20347-3

Library of Congress Control Number: 2011925863

© 2011 Springer-Verlag Berlin Heidelberg

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typeset & Cover Design: Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

To my family

Preface

Motto: *We need to develop
thinking, rather than
too much knowledge.*
Democritus

This book represents my 30 years continuing education courses for graduate and master degree students at the Electronics and Telecommunications Faculty from the Technical University of Cluj Napoca, Romania and partially my research activity too. The presented topics are useful for engineers, M.Sc. and PhD students who need basics in information theory and coding.

The work, organized in five Chapters and four Appendices, presents the fundamentals of Information Theory and Coding.

Chapter 1 (Information Transmission Systems - ITS) is the introductory part and deals with terminology and definition of an ITS in its general sense (telecommunication or storage system) and its role.

Chapter 2 (Statistical and Informational Model of an ITS) deals with the mathematical and informational modeling of the main components of a digital ITS: the source (destination) and the transmission channel (storage medium). Both memoryless and memory (Markov) sources are analyzed and illustrated with applications.

Chapter 3 (Source Coding) treats information representation codes (from the numeral system to the genetic code), lossless and lossy (DPCM and Delta) compression algorithms. The main efficiency compression parameters are defined and a detailed presentation, illustrated with many examples, of the most important compression algorithms is provided, starting with the classical Shannon-Fano or Huffman until the modern Lempel Ziv or arithmetic type.

Chapter 4 (Cryptography Basics) is presenting basics of classic and modern symmetric and public key cryptography. Introduction in DNA cryptography and in digital watermarking are ending the chapter. Examples are illustrating all the presented chipers.

Chapter 5 (Channel Coding) is the most extended part of the work dealing with error control coding: error detecting and forward error correcting codes. After defining the aim of channel coding and ways to reach it as established by Shannon second theorem, the elements of the theory of block codes are given and Hamming group codes are presented in detail.

Cyclic codes are a main part of Chapter 5. From this class, a detailed presentation of BCH, Reed-Solomon, Golay and Fire codes is given, with linear feedback shift register implementation. The algebraic decoding algorithms, Peterson and Berlekamp, are presented.

Concerning the convolutional codes, after a short comparison with block codes, a detailed description of encoding and graphical representation as well as decoding algorithms are given.

Principles of interleaving and concatenation are also presented and exemplified with the CIRC standard used in audio CD error control.

The principles of the modern and powerful Turbo Codes are ending the presentation of error control codes. Channel Coding chapter also includes a presentation of Base-Band coding.

The work also includes four Appendices (A, B, C and D) presenting: A – Algebra elements and tables concerning some Galois fields and generator polynomials of BCH and RS codes; B – Tables for information and entropy computing; C – Signal detection elements and D – Synthesis example.

I tried to reduce as much as possible the mathematical demonstrations, focusing on the conditions of theorems validity and on their interpretation. The examples were selected to be as simple as possible, but pointing out the essential aspects of the processing. Some of them are classic, others are taken from the literature, being currently standards in many real applications, but most of them are original and are based on typical examples taken from my lectures.

The understanding of the phenomenon, of the aims of processing (compression, encryption and error control) in its generality, not necessarily linked to a specific application, the criteria of selecting a solution, the development of the “technical good sense”, is the logic thread guiding the whole work.

Acknowledgements

I want to thank to all those who directly or indirectly, contributed to my training as an engineer and professor.

Special thanks are addressed to Professor Alexandru Spataru and to the school he created. He was my teacher in Theory of Information Transmission, during the time I was student at Polytechnic Institute of Bucharest.

For the interest shown towards the modern algebraic decoding algorithms, for the simplicity and practical way in which he presented the abstract mathematics, I thank professor mathematician Nicolae Ghircoiasu, now pastaway.

I am highly grateful to Dr. Graham Wade, from the University of Plymouth, UK, for the modern and practical orientation he gave to my whole course, for the valuable discussions and fruitful collaboration we had since 1993. His books impressed me a lot, due to the practical features that he gave to a domain usually presented theoretically.

Special thanks are given to my colleagues, professor Romulus Terebeş, teaching assistant Raul Malutan and researcher Bogdan Belean for their observations and suggestions, and to my PhD student Olga Tornea for her contributions to DNA cryptography.

I kindly acknowledge to all students who helped me in typewriting the book.

Chapter 3 and 4 were partially supported by the Romanian National Research Council CNCSIS-UEFISCSU project no. PNII ID_909/2007.

I would like to thank Dr. Christoph Baumann, Senior Editor Engineering at Springer (Germany), who motivated me in preparing the book. Finally, but not least, I thank Carmen Wolf, Editorial Assistant at Springer (Germany) and Ganesan Prabu from Scientific Publishing Services (India) for their wonderful help in the preparation and publication of the manuscript.

Contents

1	Information Transmission Systems.....	1
1.1	Terminology	1
1.2	Role of an ITS.....	2
1.3	Model of an ITS.....	3
	References	5
2	Statistical and Informational Model of an ITS	7
2.1	Memoryless Information Sources	7
2.2	Measure of Discrete Information	8
2.3	Information Entropy for a DMS (Shannon Entropy)	11
2.4	Source Redundancy and Efficiency	13
2.5	Entropy of an Extended Discrete Memoryless Source: $H(X_n)$	14
2.6	Moments and Moment Rate.....	14
2.7	Information Rate, Decision Rate.....	15
2.8	Discrete Transmission Channels.....	16
2.8.1	Probabilities and Entropies in Discrete Channels	16
2.8.2	Mutual Information and Transinformation.....	21
2.8.3	Relationships between Entropies	21
2.8.4	Channel Capacity Using the Noise Matrix.....	23
2.8.5	Shannon Capacity.....	30
2.9	Memory Sources (Markov Sources)	38
2.9.1	Finite and Homogeneous Markov Chains	39
2.9.2	Entropy of m-th Order Markov Source	43
2.9.3	Applications	46
	References	51
3	Source Coding.....	53
3.1	What Is Coding and Why Is It Necessary?	53
3.2	Aim of Source Coding	55
3.3	Information Representation Codes	55
3.3.1	Short History	55
3.3.2	Numeral Systems	56
3.3.3	Binary Codes Used in Data Transmission, Storage or Computing.....	58

3.3.4	Pulse Code Modulation (PCM)	65
3.3.5	Genetic Code	76
3.4	Coding Efficiency: Compression Ratio	80
3.5	Existence Theorem for Instantaneous and Uniquely Decodable Codes (Kraft and McMillan Inequalities)	82
3.6	Shannon First Theorem (1948)	84
3.7	Lossless Compression Algorithms	85
3.7.1	Shannon-Fano Binary Coding	85
3.7.2	Huffman Algorithms	88
3.7.3	Run Length Coding (RLC)	95
3.7.4	Comma Coding	100
3.7.5	Dictionary Techniques [41]	101
3.7.6	Lempel-Ziv Type Algorithms (LZ)	102
3.7.7	Arithmetic Coding	107
3.8	Lossy Compression in Differential Coding	109
3.8.1	Differential Pulse Code Modulation (DPCM)	109
3.8.2	Delta Modulation (DM)	111
3.8.3	Delta-Sigma Modulation	115
3.8.4	Comparison and Applications of Digital Modulations	116
3.9	Conclusions on Compression	116
References		118
4	Cryptography Basics	121
4.1	Short History of Cryptography	121
4.2	Terminology	123
4.3	Cryptosystems: Role and Classification	123
4.4	Cryptanalytic Attacks and Algorithms Security	125
4.5	Classic Cryptography	127
4.5.1	Definition and Classification	127
4.5.2	Caesar Cipher	128
4.5.3	Polybius Cipher	130
4.5.4	Playfair Cipher	131
4.5.5	Trithemius Cipher	132
4.5.6	Vigènère Cipher	134
4.6	Modern Symmetric (Conventional) Cryptography	135
4.6.1	Definitions and Classification	135
4.6.2	Block Ciphers	137
4.6.2.1	Main Features	137
4.6.2.2	DES (Data Encryption Standard)	139
4.6.2.3	Some Other Block Ciphers	147
4.6.2.4	Block Cipher Operation Modes	150
4.6.3	Stream Ciphers	153
4.6.3.1	General Features	153
4.6.3.2	Some Stream Ciphers	157
4.6.4	Authentication with Symmetric Cryptography	158

4.7	Public Key Cryptography	159
4.7.1	Principle of the Public Key Cryptography	159
4.7.2	Public Keys Ciphers	162
4.8	Digital Watermarking	164
4.8.1	Introduction	164
4.8.2	Short History	167
4.8.3	Watermarking Requirements	168
4.8.4	Basic Principles of Watermarking	169
4.8.5	Specific Attacks	177
4.8.5.1	Attack Definition and Classification	177
4.8.5.2	The Inversion Attack / IBM / Confusion / Deadlock / Fake Watermark / Fake Original	178
4.8.5.3	The Collusion Attack	180
4.8.6	Applications	181
4.8.6.1	Broadcast Monitoring	181
4.8.6.2	Owner Identification: Proof of Ownership	182
4.8.6.3	Fingerprinting (Transaction Tracking)	183
4.8.6.4	Content Authentication (Fragile Watermarking)	183
4.8.6.5	Copy Control	184
4.8.7	The Millennium Watermark System [53]	184
4.8.7.1	Introduction	184
4.8.7.2	Basic Principle of the Millennium System	185
4.8.7.3	Millennium Standard Implementation	188
4.8.7.4	Unsolved Problems	188
4.8.7.5	Copy Control [53]	189
4.8.7.6	Media Type Recognition	189
4.8.8	Some Remarks	189
4.9	DNA Cryptography	190
4.9.1	Introduction	190
4.9.2	Backgrounds of Biomolecular Technologies	190
4.9.3	Elements of Biomolecular Computation (BMC)	194
4.9.3.1	DNA OTP Generation	194
4.9.3.2	Conversion of Binary Data to DNA Format and Vice Versa	194
4.9.3.3	DNA Tiles and XOR with DNA Tiles	195
4.9.4	DNA Based Steganography and Cryptographic Algorithms	197
4.9.4.1	Steganography Technique Using DNA Hybridization	197
4.9.4.2	Chromosome DNA Indexing Algorithm	199
4.9.4.3	DNA XOR OTP Using Tiles	202
	References	204
5	Channel Coding	209
5.1	Shannon Second Theorem (Noisy Channels Coding Theorem)	209
5.2	Error Control Strategies	213

5.3	Classification of Error Control Codes.....	216
5.4	Representation of Binary Code Sequences	216
5.5	Parameters of Detecting and Error Correcting Codes.....	218
5.6	Maximum Likelihood Decoding (MLD)	220
5.7	Linear Block Codes	224
5.7.1	Linear Block Codes: Definition and Matrix Description	224
5.7.2	Error Syndrome.....	227
5.7.3	Dimensioning of Error Correcting Linear Block Codes.....	228
5.7.4	Perfect and almost Perfect Codes.....	228
5.7.5	Detection and Correction Capacity: Decoded BER	229
5.7.6	Relations between the Columns of H Matrix for Error Detection and Correction	230
5.7.7	Standard Array and Syndrome Decoding.....	232
5.7.8	Comparison between Linear Error Detecting and Correcting Block Codes	235
5.7.9	Hamming Group Codes.....	238
5.7.10	Some Other Linear Block Codes.....	253
5.8	Cyclic Codes.....	255
5.8.1	Definition and Representation.....	256
5.8.2	Algebraic Encoding of Cyclic Codes	257
5.8.3	Syndrome Calculation and Error Detection	265
5.8.4	Algebraic Decoding of Cyclic Codes.....	269
5.8.5	Circuits for Cyclic Encoding and Decoding.....	274
5.8.6	Cyclic One Error Correcting Hamming Code.....	286
5.8.7	Golay Code	290
5.8.8	Fire Codes	292
5.8.9	Reed–Solomon Codes	295
5.9	Convolutional Codes	312
5.9.1	Representation and Properties	312
5.9.2	Convolutional Codes Encoding.....	314
5.9.3	Graphic Representation of Convolutional Codes	320
5.9.4	Code Distance and d_{∞}	324
5.9.5	Decoding (Viterbi Algorithm, Threshold Decoding)	326
5.10	Code Interleaving and Concatenation.....	340
5.10.1	Interleaving	340
5.10.2	Concatenated Codes	342
5.11	Turbo Codes.....	346
5.11.1	Definition and Encoding	346
5.11.2	Decoding	348
5.11.2.1	Basic Principles	348
5.11.2.2	Viterbi Algorithm (VA) [46], [48]	349
5.11.2.3	Bidirectional Soft Output Viterbi Algorithm (SOVA) [46].....	357
5.11.2.4	MAP Algorithm	364
5.11.2.5	MAX-LOG-MAP Algorithm	372

5.11.2.6 LOG-MAP Algorithm	373
5.11.2.7 Comparison of Decoding Algorithms	374
5.12 Line (baseband) Codes	374
References	385
Appendix A: Algebra Elements	389
A1 Composition Laws	389
A1.1 Compositions Law Elements	389
A1.2 Stable Part	389
A1.3 Properties	389
A2 Modular Arithmetic	391
A3 Algebraic Structures	392
A3.1 Group	392
A3.2 Field	393
A3.3 Galois Field	393
A.3.3.1 Field Characteristic	394
A.3.3.2 Order of an Element	395
A4 Arithmetics of Binary Fields	395
A5 Construction of Galois Fields $GF(2^m)$	398
A6 Basic Properties of Galois Fields, $GF(2^m)$	402
A7 Matrices and Linear Equation Systems	410
A8 Vector Spaces	412
A8.1 Defining Vector Space	412
A8.2 Linear Dependency and Independency	413
A8.3 Vector Space	414
A9 Table for Primitive Polynomials of Degree k (k max = 100)	417
A10 Representative Tables for Galois Fields $GF(2^k)$	418
A11 Tables of the Generator Polynomials for BCH Codes	420
A12 Table of the Generator Polynomials for RS Codes	421
Appendix B: Tables for Information and Entropy Computing	427
B1 Table for Computing Values of $-\log_2(x)$, $0.01 \leq x \leq 0.99$	427
B2 Table for Computing Values of $-x \cdot \log_2(x)$, $0.001 \leq x \leq 0.999$	428
Appendix C: Signal Detection Elements	435
C.1 Detection Problem	435
C.2 Signal Detection Criteria	438
C.2.1 Bayes Criterion	438
C.2.2 Minimum Probability of Error Criterion (Kotelnikov- Siegert)	440
C.2.3 Maximum a Posteriori Probability Criterion (MAP)	441
C.2.4 Maximum Likelihood Criterion (R. Fisher)	441
C.3 Signal Detection in Data Processing ($K = 1$)	442
C.3.1 Discrete Detection of a Unipolar Signal	442
C.3.2 Discrete Detection of Polar Signal	446

C.3.3 Continuous Detection of Known Signal448
C.3.4 Continuous Detection of Two Known Signals453
References459
Appendix D: Synthesis Example461
Subject Index.....475
Acronyms.....483

List of Figures

1.1	Block scheme of a general digital ITS	3
1.2	Illustration of signal regeneration in digital communications: a) original signal, b) slightly distorted signal, c) distorted signal, d) intense distorted signal, e) regenerated signal (l - distance in transmission)	4
2.1	Graphical representation of the entropy corresponding to a binary source	13
2.2	Discrete information sources: a) unipolar binary source (m=2), b) polar binary source (m=2), c) quaternary source (m=4)	15
2.3	Discrete channel: a) graph representation, b) matrix representation	17
2.4	Graphical representation of the relationships between entropies: a) ordinary channel, b) noiseless channel, c) independent channel	23
2.5	The graph corresponding to a binary erasure channel	26
2.6	Graphical representation of a binary transmission system	27
2.7	Illustration of time and amplitude resolutions	31
2.8	Graphical representation of channel capacity	33
2.9	Bandwidth - capacity dependency	34
2.10	Bandwidth-efficiency diagram	35
2.11	Graphical representation of the relations between the information source and the channel	36
2.12	The graph corresponding to a Markov chain with two states	40
2.13	Transition from s_i to s_j in two steps	41
2.14	Absorbent Markov chain	42
2.15	Graph corresponding to a 2-step memory binary source	45
2.16	Differential Pulse Code Modulation for a black-and-white video signal	49
2.17	Example of burst	50
2.18	The simplified model of a memory channel	50
3.1	Illustration of the transformation $S \rightarrow X$ realized through encoding	54
3.2	Coding tree associated to code D from Table 3.1	55
3.3	Block scheme illustrating the generation of PCM signal (PCM modulator and coder)	65
3.4	Block scheme illustrating the receiving PCM process (PCM demodulator / decoder)	65
3.5	Example of PCM generation	66
3.6	Representation of quantisation noise pdf	67
3.7	Companding illustration	71
3.8	Ideal compression characteristic	72
3.9	Threshold effect in PCM systems	75

3.10	DNA structure.....	77
3.11	The coding tree of Example 3.5.....	86
3.12	Evolution of dynamic Huffman FGK tree for the message 'abcbb'	92
3.13	The effect of one error on the modified Huffman code: a) transmitted sequence, b) and c) received sequence affected by one error	98
3.14	Uniform steps coding for black and white images.....	100
3.15	Text compression.....	100
3.16	The LZ-77 algorithm illustration	103
3.17	Illustration of LZ-77 algorithm.....	104
3.18	Illustration of LZ-78 algorithm.....	104
3.19	Illustration of DPCM principle	109
3.20	DPCM codec: a) DPCM encoder; b) DPCM decoder	110
3.21	Illustration of the DM	112
3.22	Slope-overload noise	112
3.23	Illustration of granular noise.....	113
3.24	Granular noise in DM - pdf representation	114
3.25	Delta-Sigma modulation - demodulation block scheme	115
3.26	Classification of compression.....	116
4.1	Block-scheme of a cryptosystem where: A, B - entities which transmit, receive the information, E - encryption block, D - decryption block, M- plaintext, C - ciphertext, K - cryptographic key block, k_e - encryption key, k_d - decryption key	123
4.2	Illustration of confidentiality	124
4.3	Illustration of authentication.....	124
4.4	Alberti cipher disk (formula).....	132
4.5	Example of P box with $n=7$	138
4.6	Example of S box with $n = 3$: a) block scheme, b) truth table.....	138
4.7	Example of a product cipher (alternation of P and S boxes).....	139
4.8	Generation of round keys in DES	142
4.9	DES encryption routine	144
4.10	DES encryption/decryption function f	145
4.11	Illustration of whitening technique: a- encryption, b- decryption.....	149
4.12	ECB mode: a) encryption; b) decryption	150
4.13	CBC mode : a) encryption, b) decryption.....	152
4.14	CFB mode: encryption.....	152
4.15	OFB mode : encryption	153
4.16	Block scheme of a pseudo-noise generator using LFSR.....	154
4.17	Pseudo-noise generator with LFSR and $g(x)=x^3+x^2+1$	155
4.18	Pseudo noise sequences generator implemented with two LFSRs	156
4.19	Nonlinear multiplexed system for generating pseudo-noise sequences.....	157
4.20	Block scheme of a public key system.....	159
4.21	Confidentiality in public key cryptosystems.....	160
4.22	Authentication in public key cryptosystems	160
4.23	Confidentiality and authentication in public key cryptosystems	161
4.24	Watermark principle block scheme: a) insertion (embedding), b) retrieval / detection	166


4.25	Bloc scheme for watermark insertion	170
4.26	Bloc scheme for watermark extraction and comparison	171
4.27	Line - scanning video signal	173
4.28	Video sequence watermark insertion model	174
4.29	Video sequence watermark extraction model	175
4.30	Millennium Watermark block schemes: a) insertion, b) detection	186
4.31	Hybridization process	191
4.32	Gene representation	191
4.33	Chromosome representation (http://www.ohiohealth.com/)	192
4.34	Amplifying process in PCR technique	192
4.35	Illustration of DNA recombinant process	193
4.36	Illustration of microarray experiment	193
4.37	Binding process between two ssDNA segments	194
4.38	Triple-helix tile	195
4.39	Tiles assembling through complementary sticky ends	196
4.40	Tiles for START bits in a string	196
4.41	Stacks with tiles for the rest of the bits in a string	196
4.42	XOR computation with tiles	197
4.43	Cipher text hiding: a) structure of cipher text inserted between two primers; b) dsDNA containing (hidden) the cipher text	198
4.44	Illustration of OTP scanning process for message encryption	200
4.45	Design of the One Time Pad tiles	203
4.46	Example of message tiles binding	204
4.47	Design of the one-time-pad tiles	204
5.1	Error exponent graphic	210
5.2	Input/output representation of a BSC obtained for C2	211
5.3	Illustration of the relations between information bits and coded ones	212
5.4	Location of channel coding block in a complete transmission (storage) system	213
5.5	ARQ systems for $N = 5$: a) SW(Stop and Wait); b) GBN(go back N); c) SR(selective repeat)	215
5.6	Minimum distance decoder principle ($d=5$)	222
5.7	Coding gain representation	238
5.8	Shortened Hamming code for 16 bits memory protection	244
5.9	Hamming (7,4) code block scheme: a) encoding unit; b) decoding unit	247
5.10	HDLC frame format	268
5.11	LFSR with external modulo 2 adders	274
5.12	LFSR with internal modulo 2 adders	276
5.13	Cyclic systematic encoder with LFSR and external modulo two adders	277
5.14	Cyclic systematic encoder with LFSR and internal modulo two adders	279
5.15	Cyclic encoder with LFSR and external \oplus : a) block scheme; b) operation table for $g(x) = x^3 + x + 1$ and $m = 4$	280
5.16	Cyclic encoder with LFSR and internal \oplus : a) block scheme; b) operation table for $g(x) = x^3 + x + 1$ and $m = 4$	281
5.17	Error detection cyclic decoder with LFSR and external \oplus	282
5.18	Error detection cyclic decoder with LFSR and internal \oplus	283

5.19	a) Block scheme and b) the operating table of the cyclic decoder for $g(x) = x^3 + x + 1$	284
5.20	a) Block scheme and b) the operation table of a cyclic error detection decoder with LFSR and internal \oplus	285
5.21	General block scheme of an error correction cyclic decoder	286
5.22	a) SR block scheme and b) operation of the cyclic decoder from Fig. 5.21, for the cyclic Hamming code (7,4) with $g(x) = x^3 + x + 1$; LFSR with external \oplus	289
5.23	a) SR block scheme and b) operation of the cyclic decoder from Fig. 5.21, for the cyclic Hamming code (7,4) with $g(x) = x^3 + x + 1$; LFSR with internal \oplus	290
5.24	Block schemes for Fire code with $g(x) = x^{10} + x^7 + x^2 + 1$ and $p = 3$: a) encoder and b) decoder	294
5.25	Flow-chart corresponding to Berlekamp-Massey algorithm.....	307
5.26	Summating circuit over $GF(2^k)$	309
5.27	Multiplying circuit with α over $GF(2^4)$	310
5.28	CD player – block scheme	311
5.29	Comparison between block and convolutional codes	313
5.30	Block scheme of: a) systematic and b) non systematic convolutional encoder; (ISR - Information Shift Register)	315
5.31	Representation of the information at: a) encoder input; b) systematic encoder output; c) non-systematic encoder output.....	316
5.32	a) Convolutional systematic encoder - block scheme and operation; b) Convolutional non-systematic encoder - block scheme and operation	319
5.33	State diagram of the convolutional code with $R = 1/2$, $K = 3$ for a) systematic and b) non-systematic type	321
5.34	The graph corresponding to the systematic convolutional code $R = 1/2$, $K = 3$; — the encoded structure for $i = [0 \ 1 \ 1 \ 0 \ 1]$	322
5.35	Trellis corresponding to the convolutional code $R = 1/2$, $K = 3$: a) systematic with $g(x) = 1 + x^2$; b) systematic with $g(x) = 1 + x + x^2$; c) non-systematic with $g_1(x) = 1 + x^2$, $g_2(x) = 1 + x + x^2$	323
5.36	Viterbi algorithm example for non-systematic convolutional code with $R=1/2$, $K=3$, $g_1(x)=1+x^2$, $g_2(x)=1+x+x^2$, on variable number of frames: $N=3 \div 12$	328
5.37	Viterbi decoding process for a three error sequence in the first constraint length.....	332
5.38	Graphical illustration of the hard decision (with 2 levels) and the soft decision (with 8 levels)	333
5.39	Block-scheme of a threshold decoding system	334
5.40	Threshold decoder for the direct orthogonal code $R = 1/2$, $g_2(x)=1 + x^2 + x^5 + x^6$	337
5.41	Threshold decoder for indirect orthogonal code $R = 1/2$, $g_2(x)=1 + x^3 + x^4 + x^5$	339
5.42	Interleaving example: a) 4 codewords (of length 5) non-interleaved succession; b) interleaved succession ($b =$ burst error length).....	340

5.43	Block interleaving.....	341
5.44	Block-scheme of a convolutional interleaving made with shift registers (C-encoder for independent errors; dC-decoder for independent errors; I – interleaver; dI – de-interleaver)	342
5.45	Block scheme of a concatenated system.....	342
5.46	Block-scheme of an interleaved concatenated system (C ₂ - RS + C ₁ - convolutional).....	343
5.47	Block scheme of CIRC system.....	344
5.48	Encoding and interleaving in CIRC system: a) I ₁ - even samples B _{2p} are separated from the odd ones B _{2p+1} with 2Tf ; b) C ₂ - RS (28,24) encoding; c) I ₂ - samples are delayed with different time periods to spread the errors; d) C ₁ -RS (32, 28) encoding; e) I ₃ - even samples (B _{2p,i}) cross interleaving with the next frame odd samples (B _{2p+1,i+1}).....	344
5.49	Illustration of the decoding process in CIRC system.....	345
5.50	Basic RSC code: R=1/2, g ₀ - feedback polynomial, g ₁ - feed forward polynomial.....	347
5.51	Basic turbo transmitter: turbo encoder with R=1/3, BPSK modulation and AWGN channel.....	348
5.52	Basic turbo-decoder.....	349
5.53	RSC encoder with R = 1/2 and K=2: a) block- scheme; b) state-diagram; c) trellis:— i=0 input, i=1 input.....	352
5.54	Trellis representation on τ=4 frames of RSC code with R=1/2 and K=2.....	360
5.55	Forward recursion; the thick line is representing the ML path (with minimum path metric 0.04).....	362
5.56	The backward recursion; ML path is presented with thick line.....	363
5.57	MAP decoding illustration: a) block scheme of a transmission system with RSC (R = 1/2, K = 3, G = [1, (1 + x ²)/ (1 + x + x ²)]) and MAP decoding; b) state transition diagram of RSC encoder with R = 1/2, K = 3, G = [1, (1 + x ²)/ (1 + x + x ²)]; c) trellis diagram of RSC code from b)	365
5.58	Graphical MAP representation: a) forward recursion, b) backward recursion.....	369
5.59	Differential a) encoder and b) decoder.....	376
5.60	Signal detection: a) block-scheme and b) illustration of BER.....	381
5.61	Examples of Base Band encoding.....	384
C.1	Block scheme of a transmission system using signal detection. S- source, N- noise generator, SD- signal detection block, U- user, s _i (t) - transmitted signal, r(t) - received signal, n(t) - noise voltage, ŝ _i (t) - estimated signal.....	435
C.2	Binary decision splits observation space Δ into two disjoint spaces Δ ₀ and Δ ₁	436
C.3	Block scheme of an optimal receiver (operating according to Bayes criterion, of minimum risk).....	438
C.4	Binary detection parameters: P _m - probability of miss, P _D - probability of detection, P _F - probability of false detection	439
C.5	Graphical representation of function Q(y).....	440

C.6	Block-scheme of the optimal receiver for unipolar signal and discrete observation	444
C.7	Graphical representation of pdfs of decision variables for unipolar decision and discrete observation	445
C.8	Block Scheme of an optimal receiver with continuous observation decision for one known signal $s(t)$: a) correlator based implementation; b) matched filter implementation.....	451
C.9	Graphical representation of $p(z/s_0)$ and $p(z/s_1)$	452
C.10	Observation space in dimensions (r_1, r_2)	455
C.11	Block- scheme of an optimal receiver for continuous decision with two known signal (correlator based implementation).....	457
C.12	Representation of the decision process in continuous detection of two known signals	458
D.1	Block-scheme of processing where:	462
D.2	Block scheme of cyclic encoder with LFSR with external modulo two sumators and $g(x) = x^3 + x + 1$	468
D.3	Non-systematic convolutional encoder for $R=1/2$ and $K=3$ ($g^{(1)}(x) = 1 + x + x^2$, $g^{(2)}(x) = 1 + x$).....	472

List of Tables

2.1	Prefixes and multiples for bit and byte	10
2.2	Illustration of time and amplitude resolution requirements	37
3.1	Binary codes associated to a quaternary source (M=4)	54
3.2	Conversion between hex, dec, oct and binary numeral systems	58
3.3	Tree corresponding to Morse code	59
3.4	Baudot Code	60
3.5	The 7-bit ISO code (CCITT N° 5, ASCII). Command characters:  - for national symbols, SP – Space, CR - Carriage Return, LF - Line Feed, EOT - End of Transmission, ESC – Escape, DEL - Delete	61
3.6	IBM BCD code – 6 bits length	62
3.7	EBCDIC code	63
3.8	Binary natural and Gray 4 bits length codes representation.....	64
3.9	Example of 3 bit code in BN and Gray representation	64
3.10	Genetic code – encoding table	78
3.11	Genetic code – decoding table	78
3.12	Modified Huffman code used in fax	97
3.13	LZW algorithm applied to Example 3.14	106
4.1	Classification of classic ciphers	128
4.2	Caesar cipher	129
4.3	Polybius square	130
4.4	Illustration of Playfair cipher	131
4.5	Tabula recta of Trithemius cipher	133
4.6	Vigènère encryption with key word.....	134
4.7	Vigènère encryption with trial- key letter and plaintext keyword	135
4.8	Vigènère encryption with trial-key and ciphertext keyword.....	135
4.9	DNA to binary conversion.....	194
4.10	Truth table of XOR function.....	197
5.1	Standard array for a C(n,m) code.....	232
5.2	Standard array for C(5,3).....	233
5.3	Syndrome decoding for a linear code C(n,m)	234
5.5	Syndrome-decoding table for Hamming (7,4) code.....	248
5.4	Standard array for the Hamming (7,4) code	249
5.6	Encoding table for the cross parity check code.....	254
5.7	Table of primitive polynomials up to degree $k = 5$	260
5.8	BCH codes generator polynomials up to $n = 31$	261
5.9	$GF(2^4)$ generated by $p(x) = x^4 + x + 1$	263
5.10	Coefficients of the error polynomials for BCH codes	272

5.11	Cyclic encoder with LFSR and external modulo two adders	278
5.12	σ_t coefficients of the error polynomial for RS codes	299
5.13	Y_k coefficients for RS codes ($t=1,2$)	300
5.14	d_∞ for the systematic and non-systematic codes: $R = 1/2$ and $K \in [2, 8]$...	326
5.15	The most important direct orthogonal codes [47] for $R = 1/2$	338
5.16	The most important indirect orthogonal codes for $R = 1/2$	339
5.17	Example of differential coding	376
A.1	Minimal polynomials for $GF(2^3)$ and generating polynomial $1 + X + X^3$	407
A.2	Minimal polynomials for $GF(2^4)$ and generating polynomial $1 + X + X^4$	407
A.3	Minimal polynomials for $GF(2^5)$ and generating polynomial $1 + X + X^5$	408
D.1	Operation of the encoder from Fig. D.1 for the binary stream i (the output of the compression block)	473

Chapter 1

Information Transmission Systems

Motto: *When desiring to master science, nothing is worse than arrogance and more necessary than time.*
Zenon

1.1 Terminology

We call *information* “any message that brings a specification in a problem which involves a certain degree of uncertainty” [9]. The word information derived from the ancient Greek words “*eidōs*” (idea) and “*morphe*” (shape, form), have thus, the meaning of form/shape of the mind.

Taking this definition into consideration we may say that *information* is a fundamental, abstract notion, as energy in physics.

Information has sense only when involves two correspondents: one generating it (the information source *S*) and another receiving it (the destination *D*, or the user *U*). Information can be transmitted at distance or stored (memorized) for later reading. The physical medium, including the contained equipment, that achieves the remote transmission of the information from *S* to *D*, is called *transmission channel C*; in the case of storage systems the channel is replaced by the *storage medium*, e.g. CD, tape etc.

Information is an abstract notion. This is why, when stored or transmitted, it must be embedded into a physical form (current, voltage, electromagnetic wave) able to propagate through the channel or to be stored. What we call *signal* is precisely this physical embodiment carrying information.

Remark

Generally speaking, by signal we understand any physical phenomenon able to propagate itself through a medium. One should notice that this definition is restrictive: it rules out the signal that interferes with the information-carrying signal (useful signal); this signal is known as *noise* or *perturbation (N)*.

The information source can be discrete (digital source), or continuous (signal source). The discrete source generates a finite number of symbols (e.g. 0 and 1 used in digital communications) while the continuous source, an infinite number of symbols (e.g. voice, television signal, measurement and control signals).

Remark

The sources, as well the destinations, are supposed to have transducers included. By *Information Transmission System (ITS)* we will understand the ensemble of interdependent elements (blocks) that are used to transfer the information from source to destination.

Remarks

- When transmitting the information from source to a remote destination through a channel, we deal with a *transmission system*; on the other hand, when storing the information, we deal with a *storage system*. The problems met in information processing for storage are similar in many aspects to those from transmission systems; therefore in the present work the term information transmission system (ITS) will be used for the general case (transmission as well storage system).
- The signal, as well as the noise, are assumed to be random.

1.2 Role of an ITS

The role of an ITS is to ensure a high degree of fidelity for the information at destination, regardless to the imperfections and interferences occurring in the channel or storage medium. The accuracy degree is estimated using a fidelity criterion, as follows:

For analogue systems:

- *mean squared error* ε :

$$\varepsilon =: \overline{[x(t) - y(t)]^2} \quad (1.1)$$

where $x(t)$, $y(t)$ are the signals generated by the source respectively received at destination; the symbol “ $\overline{\quad}$ ” indicates the time averaging.

- *signal/noise ratio (SNR)* ξ :

$$\xi =: \frac{\overline{[y(t)]^2}}{\overline{[n(t)]^2}} \quad (1.2)$$

where $n(t)$ indicates the noise.

For digital systems:

- *bit error rate (BER)*: the probability of receiving an erroneous bit

The degree of signal processing for transmission or storage, depends on the source, destination, channel (storage medium), the required accuracy degree, and the system cost.

When the source and destination are human beings, the processing may be reduced due to the physiological thresholds (hearing and vision), and also to the human brain processing, requiring a lower degree of fidelity.

When dealing with data transmissions (machines as source and destination), the complexity of processing increases in order to achieve the required fidelity.

For high quality data transmission/storage we may as well improve the channel (storage medium), the choice of the used method being made after comparing the

price of the receiver (the equipment used for processing) with the price of the channel (storage medium). The constant decrease of the LSI and VLSI circuits prices justifies the more and more increasing complexity of the terminal equipment, the ultimate purpose being the achievement of high quality transmission/storage.

Remark

In what follows we will exclusively analyze the numerical (digital) transmission systems taking into consideration their present evolution and the future perspectives that show their absolute supremacy even for applications in which the source and the destination are analogue (e.g. digital television and telephony).

1.3 Model of an ITS

The general block scheme of an ITS is presented in Fig. 1.1 which shows the general processing involving information: coding, modulation, synchronization, detection.

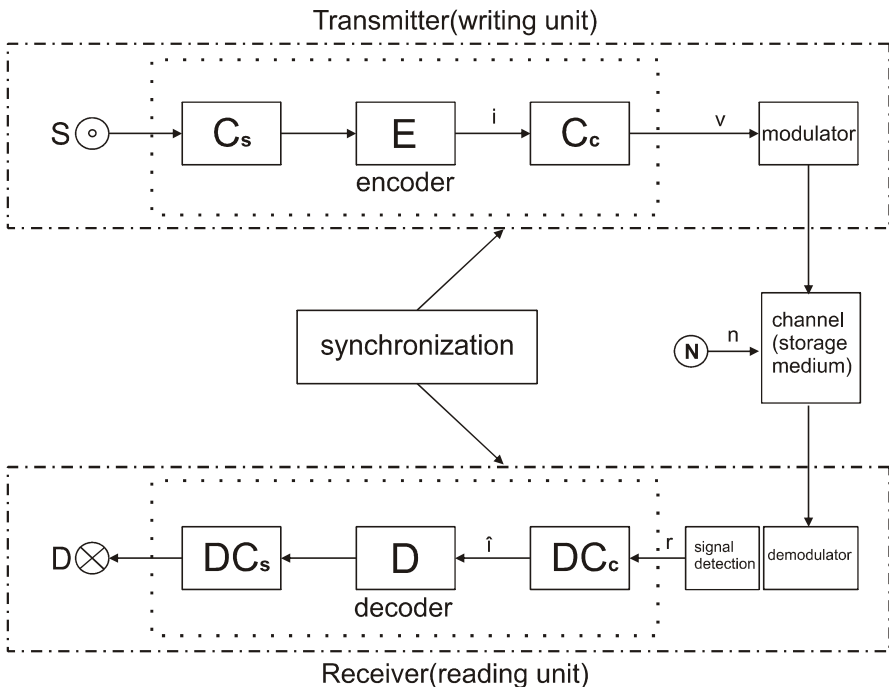


Fig. 1.1 Block scheme of a general digital ITS

Legend:

- S/D – source/destination;
- C_s/DC_s – source encoding/decoding blocks;
- E/D – source encryption/decryption blocks;

- C_C/DC_C – channel encoding/decoding blocks;
- i – information (signal)
- v – encoded word
- n – noise
- r – received signal
- \hat{i} – estimated information.

The process named *coding* stands for both encoding and decoding and is used to achieve the followings:

- matching the source to the channel/storage medium (if different as nature), using the source encoding block (C_S)
- ensuring efficiency in transmission/storage, which means minimum transmission time/minimal storage space, all these defining source compression (C_S)
- reliable transmission/storage despite channel/storage medium noise (error protection performed by the channel coding block C_C)
- preserving information secrecy from unauthorized users, using the source encryption/decryption blocks (E/D)

Modulation is used to ensure propagation, to perform multiple access and to enhance the SNR (for angle modulation), as well as to achieve bandwidth compression [8], [25].

For digital systems, *synchronization* between transmitter and receiver is necessary, and also *signal detection*, meaning that the receiver must decide, using the received signal, which of the digital signals has been sent [10].

In real applications, all the above-mentioned processes or only some of them could appear, depending on the processing degree required by the application.

Why digital?

There are several reasons why digital systems are widely used. Their main advantage is high noise immunity, explained by signal regeneration: a digital signal, having only two levels corresponding to “0” and “1”, allows an easy regeneration of the original signal, even from a badly damaged signal (fig.1.2), without accumulation of regenerative errors in transmission (in contrast to analogue transmission) [5], [18].

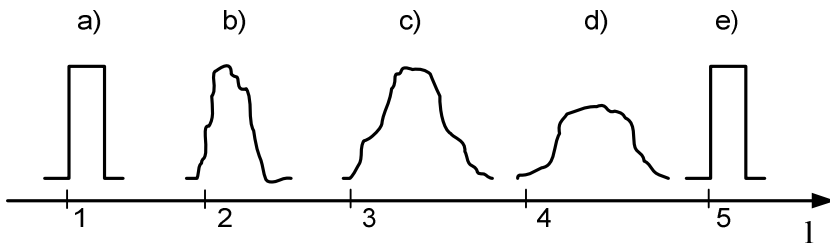


Fig. 1.2 Illustration of signal regeneration in digital communications: a) original signal, b) slightly distorted signal, c) distorted signal, d) intense distorted signal, e) regenerated signal (1 - distance in transmission).

In analogue systems, the distortions, however small, cannot be eliminated by amplifiers (repeaters), the noise accumulating during transmission; therefore in order to ensure the required fidelity for a specific application, we must use a high SNR, unlike for digital systems in which (taking into account the possibility of error protection) we may use a very low SNR (lower than 10 dB, near Shannon limit [2]).

Other advantages of digital systems are:

- possibility of more flexible implementation using LSI and VLSI technologies
- reliability and lower price than for analogue systems
- identical analysis in transmission and switching for different information sources: data, telegraph, telephone, television, measurement and control signals (the principle of ISDN – Integrated Switching Digital Network)
- good interference and jamming protection and also the possibility of ensuring information confidentiality.

The main disadvantage of digital systems is the increased bandwidth compared to analogue ones. This disadvantage can be diminished through compression as well as through modulations, for spectrum compression.

References

- [1] Angheloiu, I.: Teoria Codurilor. Editura Militara, Bucuresti (1972)
- [2] Berrou, C., Glavieux, A.: Near Shannon limit error-correcting coding and decoding turbo-codes. In: Proc. ICC 1993, Geneva, pp. 1064–1070 (1993)
- [3] Borda, M.: Teoria Transmiterii Informatiei. Editura Dacia, Cluj-Napoca (1999)
- [4] Borda, M.: Information Theory and Coding. U.T. Pres, Cluj-Napoca (2007)
- [5] Fontolliet, P.G.: Systèmes de télécommunications. Editions Georgi, Lausanne (1983)
- [6] Gallager, R.G.: Information Theory and Reliable Communication. John Wiley & Sons, Chichester (1968)
- [7] Hamming, R.: Coding and Information Theory. Prentice-Hall, Englewood Cliffs (1980)
- [8] Haykin, S.: Communication Systems, 4th edn. John Wiley & Sons, Chichester (2001)
- [9] Ionescu, D.: Codificare si coduri. Editura Tehnica, Bucuresti (1981)
- [10] Kay, S.M.: Fundamentals of Statistical Signal Processing. In: Detection Theory, vol. II. Prentice-Hall, Englewood Cliffs (1998)
- [11] Lin, S., Costello, D.: Error Control Coding. Prentice-Hall, Englewood Cliffs (1983)
- [12] Mateescu, A., Banica, I., et al.: Manualul inginerului electronist, Transmisii de date. Editura Tehnica, Bucuresti (1983)
- [13] McEliece, R.J.: The Theory of Information and Coding, 2nd edn. Cambridge University Press, Cambridge (2002)
- [14] Murgan, A.: Principiile teoriei informatiei in ingineria informatiei si a comunicatiilor. Editura Academiei, Bucuresti (1998)
- [15] Peterson, W.W., Weldon, E.J.: Error-Correcting Codes, 2nd edn. MIT Press, Cambridge (1972)
- [16] Proakis, J.: Digital Communications, 4th edn. Mc Gran-Hill (2001)

- [17] Shannon, C.E.: A Mathematical Theory Of Communication. Bell System Technical Journal 27, 379–423 (1948); Reprinted in Shannon Collected Papers, IEEE Press (1993)
- [18] Sklar, B.: Digital Communications, 2nd edn. Prentice-Hall, Englewood Cliffs (2001)
- [19] Spataru, A.: Teoria transmisiunii informatiei. Editura Didactica si Pedagogica, Bucuresti (1983)
- [20] Spataru, A.: Fondements de la theorie de la transmission de l'information. Presses Polytechniques Romandes, Lausanne (1987)
- [21] Tomasi, W.: Advanced Electronic Communications. Prentice-Hall, Englewood Cliffs (1992)
- [22] Wade, G.: Signal Coding and Processing. Cambridge University Press, Cambridge (1994)
- [23] Wade, G.: Coding Techniques. Palgrave (2000)
- [24] Wozencraft, J.W., Jacobs, I.M.: Principles of Communication Engineering. Waveland Press, Prospect Heights (1990)
- [25] Xiong, F.: Digital Modulation Techniques. Artech House, Boston (2000)

Chapter 2

Statistical and Informational Model of an ITS

Motto: *Measure is the
supreme well.
(from the wisdom of the peoples)*

2.1 Memoryless Information Sources

Let us consider a *discrete information source* that generates a number of m distinct symbols (messages). The set of all distinct symbols generated by the source forms the *source alphabet*.

A discrete source is called memoryless (*discrete memoryless source DMS*) if the emission of a symbol does not depend on the previous transmitted symbols.

The statistical model of a DMS is a discrete random variable (r.v.) X ; the values of this r.v. will be noted as x_i , $i = \overline{1, m}$. By $X=x_i$ we understand the emission of x_i from m possible.

The m symbols of a DMS constitute a *complete system of events*, hence:

$$\bigcup_{i=1}^m x_i = \Omega \quad \text{and} \quad x_i \cap x_j = \Phi, \forall i \neq j \quad (2.1)$$

In the previous formula, Ω signifies the certain event or the sample space and Φ the impossible event.

Consider $p(x_i) = p_i$ the emission probability of the symbol x_i . All these probabilities can be included in the *emission probability matrix* $P(X)$:

$$P(X) = [p_1 \cdots p_i \cdots p_m], \text{ where } \sum_{i=1}^m p_i = 1 \quad (2.2)$$

For a memoryless source we have:

$$p(x_i/x_{i-1}, x_{i-2} \dots) = p(x_i) \quad (2.3)$$

For the r.v. X , which represents the statistical model for a DMS, we have the *probability mass function* (PMF) of r.v. X :

$$X: \left(\begin{matrix} x_i \\ p_i \end{matrix} \right), i = \overline{1, m}, \sum_{i=1}^m p_i = 1 \quad (2.4)$$

Starting from a DMS, X , we can obtain a new source, having messages which are sequences of n symbols of the initial source X . This new source, X^n , is called the n -th order extension of the source X .

$$\begin{aligned} X &: \left(\begin{array}{c} x_i \\ p_i \end{array} \right), i = \overline{1, m}, \sum_{i=1}^m p_i = 1 \\ X^n &: \left(\begin{array}{c} m_j \\ p_j \end{array} \right), j = \overline{1, m^n}, \sum_{j=1}^{m^n} p_j = 1 \\ \text{where } &\begin{cases} m_j = x_{j_1} x_{j_2} \dots x_{j_n} \\ p_j = p(x_{j_1}) p(x_{j_2}) \dots p(x_{j_n}) \end{cases} \end{aligned} \quad (2.5)$$

The source X^n contains a number of m^n distinct m_j messages formed with alphabet X .

Example 2.1

Consider a binary memoryless source X :

$$X : \left(\begin{array}{cc} x_1 & x_2 \\ p_1 & p_2 \end{array} \right), p_1 + p_2 = 1$$

The 2nd order extension ($n=2$) of the binary source X is:

$$X^2 : \left(\begin{array}{cccc} x_1 x_1 & x_1 x_2 & x_2 x_1 & x_2 x_2 \\ p_1^2 & p_1 p_2 & p_2 p_1 & p_2^2 \end{array} \right) = \left(\begin{array}{cccc} m_1 & m_2 & m_3 & m_4 \\ * & * & * & * \\ p_1 & p_2 & p_3 & p_4 \end{array} \right), \sum_{j=1}^4 p_j^* = 1$$

2.2 Measure of Discrete Information

As shown in 1.1, information is conditioned by uncertainty (non-determination).

Consider the DMS, $X : \left(\begin{array}{c} x_i \\ p_i \end{array} \right), i = \overline{1, m}, \sum_{i=1}^m p_i = 1$. Prior to the emission of a symbol x_i there is an uncertainty regarding its occurrence. After the emission of x_i this uncertainty disappears, resulting the information about the emitted symbol. Information and uncertainty are closely connected, but not identical. They are inversely proportional measures, information being a removed uncertainty. It results that the information varies oppositely to the uncertainty.

The uncertainty regarding the occurrence of x_i depends on its occurrence probability: p_i , the both measures being connected through a function $F(p_i)$ that increases as p_i decreases.

Defining the information $i(x_i)$ as a measure of the a priori uncertainty regarding the realization of x_i , we can write:

$$i(x_i) = F(p_i) \quad (2.6)$$

In order to find a formula for the function F we impose that F carries all the properties that the information must have:

- information is positive or at least equal to 0 ($F(p) \geq 0$); we cannot get any information if the event is certain ($p=1$) or impossible ($p=0$)
- information is additive: consider an event x_i composed of two independent events x_{i1} and x_{i2} ; we have $x_i = x_{i1} \cap x_{i2}$; based on the additivity of information:

$$i(x_i) = i(x_{i1}) + i(x_{i2}) \text{ and therefore} \\ F[p(x_i)] = F[p(x_{i1})] + F[p(x_{i2})] \quad (2.7)$$

Due to the fact that x_{i1} and x_{i2} are independent events, we have:

$$p(x_{i1} \cap x_{i2}) = p(x_{i1})p(x_{i2})$$

Relation (2.7) becomes:

$$F[p(x_{i1})p(x_{i2})] = F[p(x_{i1})] + F[p(x_{i2})] \quad (2.8)$$

Taking into account the positivity of information, we obtain for the functional equation (2.8) the solution:

$$F(p_i) = -\lambda \log p_i = i(x_i) \quad (2.9)$$

where λ is a positive constant.

The information provided by relation (2.9) is called *self information* of x_i .

Units of information

The unit of information was defined starting from the simplest choice, that of choosing one from two equally probable:

$$X : \begin{pmatrix} x_1 & x_2 \\ 1/2 & 1/2 \end{pmatrix}, i(x_1) = i(x_2) = -\lambda \log \frac{1}{2} = 1$$

If the logarithm is in base 2 and $\lambda=1$ we have:

$$i(x_1) = i(x_2) = \log_2 \frac{1}{2} = 1 \text{ bit} \quad (2.10)$$

The unit previously defined is known as *bit*. According to this, relation (2.10) becomes:

$$i(x_i) = -\log_2 p_i \quad (2.11)$$

The name of *bit* comes from the contraction of the words *binary digit*, made in 1947 by J.W.Tuckey in a Bell Labs memorial and in 1948, C.I.E.Shannon first used the term in his famous paper "A Mathematical Theory of Communications" [31]. Apart from \log_2 , other bases have been used at the beginnings of the information theory: e and 10. In these cases the units are:

$$-\ln \frac{1}{e} = \ln e = 1 \text{ nit} = 1 \text{ natural unit (Hartley)}$$

and represents the choice of 1 from e; this name was given in the memory of R. V. Hartley who first introduced the notion of information measure (1928).

$$- \lg \frac{1}{10} = \lg 10 = 1 \text{ dit} = 1 \text{ decimal unit,}$$

representing the choice of 1 from 10.

The relations between these three units, using the logarithm base conversion: $\log_a x = \log_a b \cdot \log_b x$, are:

$$\begin{aligned} 1 \text{ nit} &= 1.44 \text{ bits} \\ 1 \text{ dit} &= 3.32 \text{ bits} \end{aligned}$$

In computing and telecommunications, another unit for information is commonly used: the *byte* (B), introduced in 1956 by W. Buchholz from IBM. A byte is an ordered set of bits, historically being the number of bits (typically 6, 7, 8 or 9), used to encode a character text in computer. The modern standard is 8 bits (byte).

$$1 \text{ byte} = 1 \text{ B} = 8 \text{ bits}$$

Prefixes and multiples for bit and byte are given in Tab 2.1.

Table 2.1 Prefixes and multiples for bit and byte

Prefixes for bit and byte multiples				
Decimal		Binary		
Value	SI	Value	IEC	JEDEC
1000	K kilo	1024	Ki kibi	K kilo
1000 ²	M mega	1024 ²	Mi mebi	M mega
1000 ³	G giga	1024 ³	Gi gibi	G giga
1000 ⁴	T tera	1024 ⁴	Ti tebi	
1000 ⁵	P peta	1024 ⁵	Pi pebi	
1000 ⁶	E exa	1024 ⁶	Ei exbi	
1000 ⁷	Z zetta	1024 ⁷	Zi zebi	
1000 ⁸	Y yotta	1024 ⁸	Yi yobi	

where the designation of the used acronyms is:

SI- International System of Units

IEC- International Electrotechnical Commission

JEDEC- Joint Electronic Device Engineering Council (the voice of semiconductor industry)

The binary multiples of bit and byte are linked to $2^{10} = 1024$, because digital systems are based on multiples of power of 2. However its usage was discouraged by the major standard organisations and a new prefix system was defined by IEC, which defines kibi, mebi, etc., for binary multiples.

2.3 Information Entropy for a DMS (Shannon Entropy)

The self information corresponding to a symbol x_i can be computed using relation (2.11) as we have already seen. The average information per emitted symbol is called *information entropy* and is denoted with $H(X)$:

$$H(X) = E\{i(x_i)\} = \sum_{i=1}^m p_i i(x_i) = -\sum_{i=1}^m p_i \log_2 p_i \quad (2.12)$$

where E designates the average operator (expected value) of the self information of r.v. X .

The entropy $H(X)$ represents the average uncertainty that a priori exists concerning the emission.

Remarks

- Equation (2.12), given by Claude E. Shannon in 1948 in his paper “A Mathematical Theory of Communication”, has a perfect analogy with the entropy in thermodynamics, established by Boltzmann and J.W.Gibbs in 1870, therefore it was called information entropy.
- Boltzmann-Gibbs formula represents the probabilistic interpretation of the second principle of thermodynamics:

$$S = -k \sum_{i=1}^n p_i \log p_i \quad (2.13)$$

where $k=1.38 \cdot 10^{-23}$ J/K is Boltzmann constant, and p_i are the system probabilities to be in the micro state i taken from an equilibrium ensemble of n possible. The thermodynamic entropy expresses the disorder degree of the particles in a physical system.

Shannon formula indicates the system non-determination degree from an informational point of view.

- A. N. Kolmogorov [12] showed that the mathematical expression of the information entropy is identical as form with the entropy in physics, but it would be an exaggeration to consider that all the theories from physics related to entropy contain, by themselves, elements of the information theory. Both notions have in common the fact that they measure the non-determination degree of a system, but their applications can be found in completely different spheres of knowledge.
 - Relation (2.12) is a quantitative expression of the information. Besides the quantitative aspect, information has qualitative aspects as well, not emphasized in Shannon formula, very important for applications in the area of artificial intelligence [26].
 - Formula (2.12) was given under some hypotheses: the source is a DMS, so from a mathematical point of view we have a complete system of events and there is a final equilibrium of states (events). These conditions are usually fulfilled in technical systems that have been used as models by

Shannon, but are not always achieved in the biological, social, economic systems; therefore the information theory must be used with extreme care.

Properties of the entropy

- The entropy is a continuous function with respect to each variable p_i because it is a sum of continuous functions.
- Additivity: self information is additive and so it is $H(X)$, which represents the mean value of self information.
- Entropy is maximum when the symbols have equal probabilities; this maximum value is also called *decision quantity*, $D(X)$:

$$H_{\max}(X) =: D(X) = \log_2 m \quad (2.14)$$

- Symmetry: $H(X)$ is unchanged if the events x_i are reordered.

The maximum value of the entropy can be obtained calculating the maximum value of the function (2.12) with the constraint:

$$\sum_{i=1}^m p_i = 1$$

Using the Lagrange multipliers method it results:

$$\max H(X) = \max \left\{ \Phi(p_i) = -\sum_{i=1}^m p_i \log_2 p_i + \lambda \left(\sum_{i=1}^m p_i - 1 \right) \right\} \quad (2.15)$$

The maximum value of the function $F(p_i)$ is achieved for:

$$\frac{\partial \Phi(p_i)}{\partial p_i} = 0, \forall i = \overline{1, m}, \quad (2.16)$$

$$\begin{cases} \frac{\partial \Phi(p_i)}{\partial p_i} = -\log_2 p_i - \log_2 e + \lambda = 0 \\ \frac{\partial \Phi(p_j)}{\partial p_j} = -\log_2 p_j - \log_2 e + \lambda = 0 \end{cases}$$

It follows that $\log_2 p_i = \log_2 p_j$, hence $p_i = p_j, \forall i = \overline{1, m}$. Therefore, $H_{\max}(X)$ is obtained if:

$$p_1 = p_2 = \dots = p_m = \frac{1}{m}$$

and so

$$H_{\max}(X) = D(X) = \log_2 m$$

where $D(X)$ is the decision quantity of X .

This result can be deduced also intuitively; the average uncertainty corresponding to the transmission of a symbol is maximum when all the m symbols are equally probable, so when it is the hardest to predict which symbol will be transmitted.

Example 2.2

Compute and draw the entropy of a memoryless binary source:

$$X: \begin{pmatrix} 0 & 1 \\ p & 1-p \end{pmatrix}, H(X) = -p \log_2 p - (1-p) \log_2 (1-p) \quad (2.17)$$

The graphical representation of the entropy of a binary memoryless source is given in Fig. 2.1.

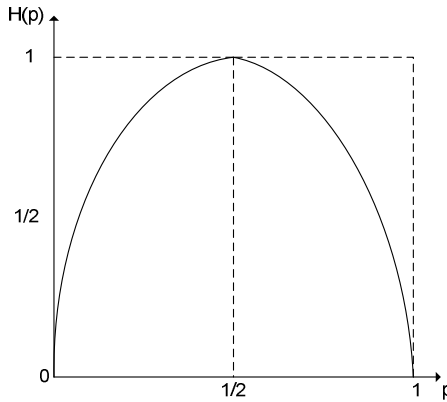


Fig. 2.1 Graphical representation of the entropy corresponding to a binary source

From Fig 2.1 we can see that $H(X)_{\max} = 1$ bit/symbol is obtained for $p = 1/2$. For $p \neq 1/2$, $H(X) < 1$; for $p=0$ or $p=1$, $H(X)=0$, because in these cases the a priori uncertainty is zero, being known exactly that 1 will be emitted in the first situation and 0 in the second, meaning that the information in these cases is zero.

2.4 Source Redundancy and Efficiency

The source entropy deviation from its maximum value is called *source redundancy*. This deviation can be absolute or relative, resulting an absolute or relative redundancy:

Absolute redundancy:

$$R_X =: D(X) - H(X) \quad (2.18)$$

Relative redundancy:

$$\rho_X =: \frac{R(X)}{D(X)} = 1 - \frac{H(X)}{D(X)} \quad (2.19)$$

Source efficiency (η_X) is the ratio between the source entropy and its decision quantity:

$$\eta_X = \frac{H(X)}{D(X)} \quad (2.20)$$

2.5 Entropy of an Extended Discrete Memoryless Source: $H(X^n)$

We consider the source X^n given by (2.5).

The entropy of this source, $H(X^n)$, is given by:

$$\begin{aligned} H(X^n) &= -\sum_{j=1}^{m^n} p_j \log_2 p_j = -\sum_{j=1}^{m^n} p_{j_1} p_{j_2} \cdots p_{j_n} \log_2 p_{j_1} p_{j_2} \cdots p_{j_n} = \\ &= -\sum_{i=1}^{m^n} p_{j_1} p_{j_2} \cdots p_{j_n} \log_2 p_{j_i} - \cdots - \sum_{i=1}^{m^n} p_{j_1} p_{j_2} \cdots p_{j_n} \log_2 p_{j_n} \end{aligned} \quad (2.21)$$

in which:

$$\begin{aligned} -\sum_{j=1}^{m^n} p_j \log_2 p_j &= -\sum_{j_1=1}^m \sum_{j_2=1}^m \cdots \sum_{j_n=1}^m p_{j_1} p_{j_2} \cdots p_{j_n} \log_2 p_{j_i} = \\ &= -\sum_{j_i=1}^m p_{j_i} \log_2 p_{j_i} \sum_{j_1=1}^m p_{j_1} \sum_{j_2=1}^m p_{j_2} \cdots \sum_{j_m=1}^m p_{j_m} = -\sum_{j_i=1}^m p_{j_i} \log_2 p_{j_i} \log_2 p_{j_i} = H(S) \end{aligned} \quad (2.22)$$

Relation (2.21) contains n (2.22) terms, hence:

$$H(X^n) = nH(X) \quad (2.23)$$

Example 2.3

The entropy of the second order extension of a memoryless binary source.

Be the binary source from Example 2.1 and its X^2 extension. Applying (2.12) for X^2 we obtain:

$$\begin{aligned} H(X^2) &= -(p_1^2 \log_2 p_1^2 + 2p_1 p_2 \log_2 p_1 p_2 + p_2^2 \log_2 p_2^2) = \\ &= -2(p_1 + p_2) \cdot (p_1 \log_2 p_1 + p_2 \log_2 p_2) = 2H(X). \end{aligned}$$

2.6 Moments and Moment Rate

The signals used for carrying the numerical information are composed of time elementary signals, called *moments*.

The characteristic parameter of any moment (amplitude, frequency, phase), remains constant during the *moment duration* (T_M) and represents the numerical information carried by that moment. This parameter can take m values.

Fig 2.2 shows some examples of information sources emphasizing the moments.

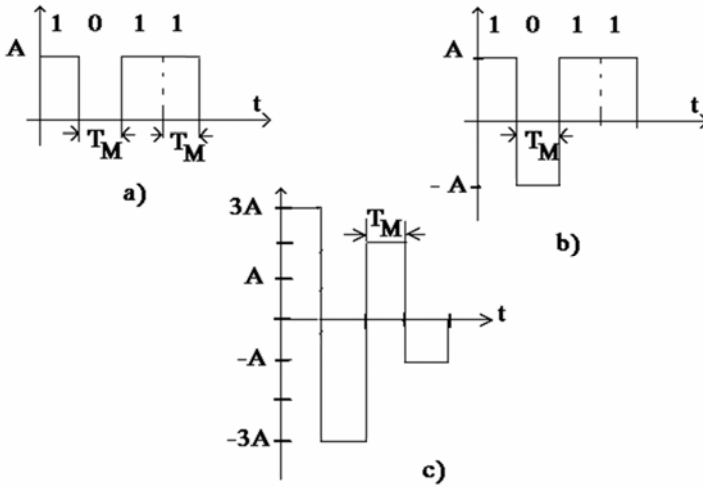


Fig. 2.2 Discrete information sources: a) unipolar binary source ($m=2$), b) polar binary source ($m=2$), c) quaternary source ($m=4$).

The decision quantity corresponding to a moment is:

$$D = \text{ld } m \tag{2.24}$$

The *moment rate* \dot{M} (*signalling speed / modulation speed / telegraphic speed*) represents the number of moments transmitted in each time unit.

$$\dot{M} =: \frac{1}{T_M} \tag{2.25}$$

The unit for the moment rate is *Baud* (Bd), named after E. Baudot, the inventor of Baudot code for telegraphy.

$$[M] = \text{Bd}$$

2.7 Information Rate, Decision Rate

Information rate $\dot{H}(X)$ of a source is the average information quantity generated by the source in time unit, or the information transmission speed.

$$\dot{H}(X) =: \frac{H(X)}{T_M} = M \dot{H}(X) \tag{2.26}$$

and its unit is:

$$\dot{H}(X) = \text{bits/second}$$

Decision rate $\dot{D}(X)$ of a source, also known, as *bit rate*, is the decision quantity generated by the source in time unit.

$$\dot{D}(X) = \frac{\dot{D}(X)}{T_M} = \dot{M} D(X) = \dot{M} \log_2 m \quad (2.27)$$

Remarks

- \dot{M} expresses the speed changes of the signal physical parameters. As we will see in the 2.8, it is directly linked to the required channel bandwidth: $\dot{M} \sim B$.
- $\dot{D} = \dot{M}$ only for binary sources ($m=2$). From (2.25) it can be seen that we can obtain the same \dot{D} in two cases: working with a higher \dot{M} and a smaller m (high speed, reduced alphabet) or inversely.

2.8 Discrete Transmission Channels

2.8.1 Probabilities and Entropies in Discrete Channels

As shown in 1.1 the transmission channel is the medium (including the equipment) used for transmitting information from the source (transmitter) to the destination (receiver) (Fig 1.1).

The *channel* is *discrete* if the symbols that are passing through it are discrete. A transmission channel is characterized by the followings:

- *input (transmission) alphabet*: $X = \{x_i\}$, is the set of distinct symbols transmitted by the source and which are accepted by the channel; $P(X)$ is the *transmission probability matrix*.

$$X : \begin{pmatrix} x_i \\ p_i \end{pmatrix}, i = \overline{1, m}, \sum_{i=1}^m p_i = 1 \quad (2.28)$$

$$P(X) = [p_i]$$

- *output(receiving) alphabet*: $Y = \{y_j\}$, being the set of all distinct symbols received at channel output; one should notice that the two alphabets are not always identical.

$$Y : \begin{pmatrix} y_j \\ q_j \end{pmatrix}, j = \overline{1, n}, \sum_{j=1}^n q_j = 1 \quad (2.29)$$

$$P(Y) = [q_j]$$

The probability of receiving the symbol y_j is $q_j=p(y_j)$. $P(Y)$ is the *reception probability matrix*.

- the *transition (noise) matrix* of the channel, containing the conditional probabilities: $P(Y/X)$.

$$P(Y/X) = [q_{j/i}], \sum_{i=1}^n q_{j/i} = 1, \forall i = \overline{1, m} \tag{2.30}$$

The element $q_{j/i}$ situated at the intersection of row i and column j , represents the reception probability of receiving y_j conditioned by the emission of x_i : $q_{j/i} = p(y_j/x_i)$.

The transition matrix is a stochastic matrix, meaning that the sum of the elements on any row is 1:

$$\sum_{j=1}^n q_{j/i} = 1, \forall i = \overline{1, m} \tag{2.31}$$

which intuitively represents the certainty in receiving a symbol $y_j, \forall j = \overline{1, n}$ if x_i was emitted, $\forall i = \overline{1, m}$.

The matrix $P(Y/X)$ represents the statistical model of the channel and is obtained experimentally.

Therefore the channel achieves the transition $[X] \rightarrow [Y]$:

$$P(X) \rightarrow P(Y/X) \rightarrow P(Y)$$

The graph in Fig 2.3 presents this transition:

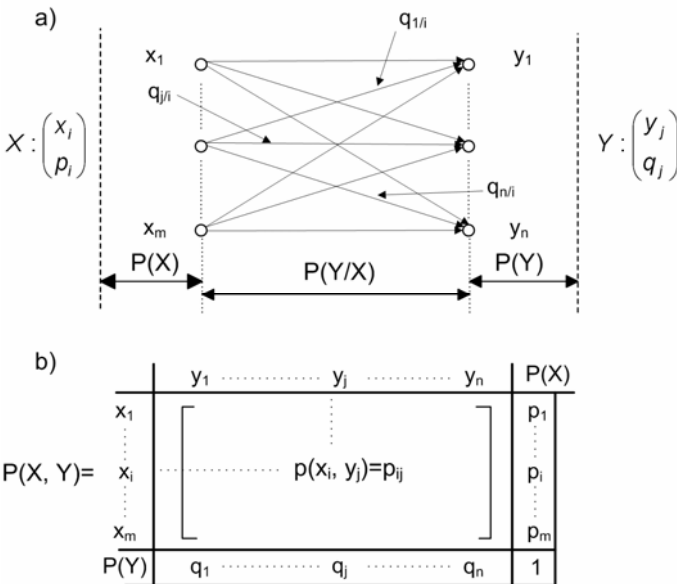


Fig. 2.3 Discrete channel: a) graph representation, b) matrix representation

If we the source is known by $P(X)$ and the channel by its noise matrix $P(Y/X)$, the receiving is found as:

$$P(Y) = P(X) P(Y/X) \quad (2.32)$$

In the previous relations the matrices $P(X)$, $P(Y)$, $P(Y/X)$ have the dimensions as given in (2.28), (2.29), (2.30) respectively.

One should notice that $P(Y)$ depends on the source through $P(X)$ and on the channel through $P(Y/X)$.

Having received the symbol y_j and knowing $P(X)$ and $P(Y/X)$, all the a-posteriori probabilities of the input symbols conditioned by y_j may be computed, using Bayes formula:

$$p(x_i/y_j) = p_{ij} = \frac{p(x_i)p(y_j/x_i)}{p(y_j)} = \frac{p_i q_{j/i}}{q_j} \quad (2.33)$$

All p_{ij} probabilities will give the conditional probability matrix $P(X/Y)$

$$P(X/Y) = [p_{ij}] = \begin{bmatrix} p(x_1/y_1) & p(x_1/y_2) & \dots & p(x_1/y_n) \\ \vdots & & & \\ p(x_m/y_1) & p(x_m/y_2) & \dots & p(x_m/y_n) \end{bmatrix} \quad (2.34)$$

Remark

The matrix $P(X/Y)$ is obtained by calculus, unlike $P(Y/X)$ which is determined experimentally and represents the channel model.

The input (X) and the output (Y) of a channel can be seen as a joint r.v. (XY): the *joint input-output*, described by the *joint probability mass function* $P(XY)$.

$P(X/Y)$ can be computed from $P(X)$ and $P(Y/X)$ using the relation:

$$\begin{aligned} P(XY) &= P(X)P(Y/X) = \\ &= \begin{bmatrix} p_1 & \dots & 0 \\ 0 & p_i & 0 \\ 0 & \dots & p_m \end{bmatrix} \cdot \begin{bmatrix} q_{1/1} & \dots & q_{n/1} \\ \dots & q_{j/i} & \dots \\ q_{1/m} & \dots & q_{n/m} \end{bmatrix} = \begin{bmatrix} p_{11} & \dots & p_{1n} \\ \dots & p_{ij} & \dots \\ p_{m1} & \dots & p_{mn} \end{bmatrix} \end{aligned} \quad (2.35)$$

Remark

In (2.35), unlike (2.32), $P(X)$ is written as a diagonal matrix. Multiplying the two matrices from (2.35), we obtain:

$$p(x_i, y_j) = p_{ij} = p_i q_{j/i} \quad (2.36)$$

Using the joint (XY) PMF, we can calculate the PMF of r.v. X and Y as marginal probabilities, as shown in Fig. 2.3.b.

$$p_i = \sum_{j=1}^n p_{ij}, \quad \forall i = \overline{1, m} \quad (2.37)$$

$$q_j = \sum_{i=1}^m p_{ij}, \forall j = \overline{1, n} \quad (2.38)$$

The five probability matrices used for statistical description of the channel will generate the corresponding entropies:

- *input entropy*:

$$P(X) \rightarrow H(X) = - \sum_{i=1}^m p_i \log_2 p_i \quad (2.39)$$

- *output entropy*:

$$P(X) \rightarrow H(X) = - \sum_{i=1}^m q_i \log_2 q_i \quad (2.40)$$

- *conditional entropy (output conditioned by input) / average error*:

$$P(Y/X) \rightarrow H(Y/X)$$

- *conditional entropy (input conditioned by output) / equivocation*:

$$P(X/Y) \rightarrow H(X/Y)$$

- *joint input – output entropy*:

$$P(XY) \rightarrow H(XY) = - \sum_{i=1}^m \sum_{j=1}^n p_{ij} \log_2 p_{ij} \quad (2.41)$$

Knowing the received symbols y_j , we cannot state that the uncertainty regarding the transmitted symbols was totally eliminated, because of the channel noise. Uncertainty still exists even after receiving the y_j symbols. The average value of this residual uncertainty is denoted by $H(X/Y)$ and signifies the input conditioned by output entropy; it is also called equivocation, being a measure of the equivocal that still exists regarding the input, when the output is known.

The information quantity obtained about x_i , when y_j has been received is, according to (2.11):

$$i(x_i/y_j) = -\log_2 p_{ij} \quad (2.42)$$

The average quantity of information obtained about the input, when y_j was received will be:

$$H(X/y_j) = \sum_{i=1}^m p_{ij} i(x_i/y_j) = - \sum_{i=1}^m p_{ij} \log_2 p_{ij}$$

The average quantity of information obtained about the input X , when we know the whole output Y , is:

$$H(X/Y) = \sum_{j=1}^n q_j H(X/y_j) = - \sum_{i=1}^m \sum_{j=1}^n q_j p_{ij} \log_2 p_{ij} ,$$

which, by taking into consideration (2.36) becomes:

$$H(X/Y) = - \sum_{i=1}^m \sum_{j=1}^n p_{ij} \log_2 p_{ij} \quad (2.43)$$

Knowing the input symbols $P(X)$, we cannot know with certainty the symbols that will be received, due to channel noise. It will always be an uncertainty whose average value is denoted with $H(Y/X)$ and it represents the output conditioned by input entropy; it is also called average error.

Using a similar reasoning with the one used to deduce (2.43), we obtain:

$$H(Y/X) = - \sum_{i=1}^m \sum_{j=1}^n p_{ij} \log_2 q_{j/i} \quad (2.44)$$

In the case of a *noiseless channel*, i.e. no interference or perturbation, the structure of the noise matrix is:

$$P(Y/X) = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & & & \\ 0 & 0 & \dots & 0 \end{bmatrix}, \quad (2.45)$$

having only 0 and 1 as elements; when we transmit the symbol x_i we know with certainty the received symbol. As a result:

$$\begin{cases} H(X/Y) = 0 \\ H(Y/X) = 0 \end{cases} \quad (2.46)$$

For a *very noisy channel (independent)*, no relation can be established between transmission and receiver, these being independent:

$$\begin{cases} p_{ij} = p_i \\ q_{j/i} = q_j \end{cases} \quad (2.47)$$

It follows that:

$$\begin{cases} H(X/Y) = H(X) \\ H(Y/X) = H(Y) \end{cases} \quad (2.48)$$

For a real channel, when the noise exists, but is not so high, the entropies will have values between the two extreme limits:

$$\begin{cases} 0 \leq H(X/Y) \leq H(X) \\ 0 \leq H(Y/X) \leq H(Y) \end{cases} \quad (2.49)$$

2.8.2 Mutual Information and Transinformation

Mutual information between x_i and y_j , denoted with $i(x_i; y_j)$, represents the non-determination that remains about the transmission of x_i after receiving y_j .

$$i(x_i; y_j) = i(x_i) - i(x_i/y_j) \quad (2.50)$$

The a priori non-determination about the transmission of x_i is $i(x_i)$. The receiving of y_j removes part of this non-determination: $i(x_i/y_j)$, the difference, given by the (2.50), being the mutual information.

Replacing in (2.50) the corresponding expressions of $i(x_i)$, respectively $i(x_i/y_j)$, and taking into consideration (2.33), we obtain:

$$i(x_i; y_j) = -\log_2 p_i + \log_2 p_{i/j} = \log_2 \frac{p_{i/j}}{p_i} = \log_2 \frac{p_{ij}}{p_i q_j} \quad (2.51)$$

The mutual information between x_i and y_j is *reciprocal*:

$$i(x_i; y_j) = i(y_j; x_i) \quad (2.52)$$

We invite the reader, as an exercise, to demonstrate the relation (2.52).

The average value of the mutual information is called *transinformation* and it is denoted with $I(X; Y)$; it represents the useful average quantity of information transmitted through the channel:

$$I(X; Y) = \sum_{i=1}^m \sum_{j=1}^n p_{ij} \log_2 \frac{p_{ij}}{p_i q_j} \quad (2.53)$$

Remark

Even though $i(x_i; y_j)$ can be negative, $I(X; Y)$ is always positive ($I(X; Y) \geq 0$).

2.8.3 Relationships between Entropies

Relation (2.41) is:

$$H(XY) = - \sum_{i=1}^m \sum_{j=1}^n p_{ij} \log_2 p_{ij}$$

Using (2.36) and (2.41), we obtain:

$$\begin{aligned} H(XY) &= - \sum_{i=1}^m \sum_{j=1}^n p_{ij} \log_2 p_{ij} q_{j/i} = - \sum_{i=1}^m \left(\sum_{j=1}^n p_{ij} \right) \log_2 p_i - \sum_{i=1}^m \sum_{j=1}^n p_{ij} \log_2 q_{j/i} = \\ &= - \sum_{i=1}^m p_i \log_2 p_i - \sum_{i=1}^m \sum_{j=1}^n p_{ij} \log_2 q_{j/i} \end{aligned}$$

Consequently, we obtain the following relation:

$$H(XY)=H(X)+H(Y/X) \quad (2.54)$$

Using a similar procedure, we obtain the result:

$$H(XY)=H(Y)+H(X/Y) \quad (2.55)$$

Remark

The reader is invited to demonstrate the following relations (2.55), (2.57), (2.58).

From (2.53) we obtain:

$$I(X; Y) = \sum_{i=1}^m \sum_{j=1}^n p_{ij} \log_2 p_{ij} - \sum_{i=1}^m \left(\sum_{j=1}^n p_{ij} \right) \log_2 p_i - \sum_{j=1}^n \left(\sum_{i=1}^m p_{ij} \right) \log_2 q_j$$

$$I(X; Y) = H(X) + H(Y) - H(X, Y) \quad (2.56)$$

$$I(X; Y) = H(X) - H(X/Y) \quad (2.57)$$

$$I(X; Y) = H(Y) - H(Y/X) \quad (2.58)$$

The relationships established between different entropies have particular expressions for distinct types of channel:

- *Noiseless channel*; from (2.46) and (2.57) we obtain:

$$I(X; Y) = H(X) = H(Y) \quad (2.59)$$

- *Independent channel*; from (2.48), (2.55), (2.57), (2.58) we get:

$$H(X, Y) = H(X) + H(Y) \quad (2.60)$$

$$I(X; Y) = 0 \quad (2.61)$$

The relationships between entropies can be graphically represented using Venns diagram:

We associate to the input r.v. X , the set A and to the output r.v. Y , the set B . We define the measure of the two sets: $m(A)$; respectively $m(B)$ as the area of these sets. Now we make the following correspondences:

- $m(A) \rightarrow H(X)$
- $m(B) \rightarrow H(Y)$
- $m(A \cup B) \rightarrow H(XY)$
- $m(A \cap \bar{B}) \rightarrow H(X/Y)$
- $m(\bar{A} \cap B) \rightarrow H(Y/X)$
- $m(A \cap B) \rightarrow I(X, Y)$

which are given in Fig. 2.4.

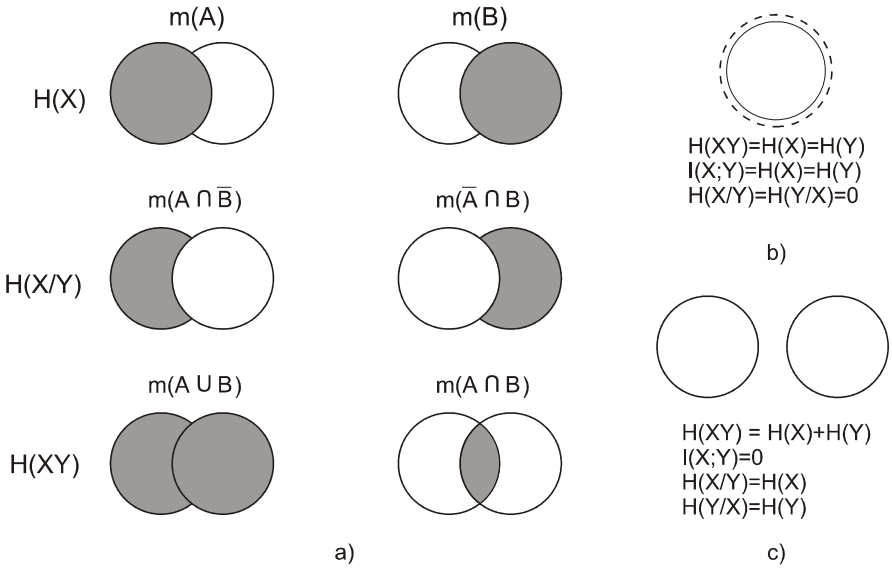


Fig. 2.4 Graphical representation of the relationships between entropies: a) ordinary channel, b) noiseless channel, c) independent channel

2.8.4 Channel Capacity Using the Noise Matrix

In order to find a measure of efficiency in transmitting the information through a channel, Cl.E. Shannon introduced the concept of *channel capacity*, C :

$$C = \max_{P_i \rightarrow P_{i0}} I(X; Y) \text{ [bits/symbol]} \tag{2.62}$$

The maximum value of the transinformation is obtained for a certain set of probabilities $[p_{i0}]$ that defines a secondary source, which will be the channel input.

Therefore, in order to transmit the maximum transinformation (which is, in fact the channel capacity) through the channel, it is compulsory that the primary source be transformed (through encoding) into a secondary source according to the optimal PMF: $[p_{i0}]$; which maximize (2.62). The entire process is called *statistical adaptation of the source to the channel*.

The maximum transinformation rate that can be transmitted through the channel is:

$$C_t = \frac{C}{T_M}; \text{ [bits/s]} \tag{2.63}$$

Remark

In many cases the channel capacity has the meaning of maximum information rate that can be transmitted through the channel ($C_t \rightarrow C$).

Channel redundancy and efficiency

By analogy with source redundancy and efficiency we can define the followings:

- *channel redundancy*, which expresses the deviation of the transinformation from its maximum value, in relative or absolute value:
- *channel absolute redundancy*, R_C :

$$R_C =: C - I(X; Y) \quad (2.64)$$

- *channel relative redundancy*, ρ_C :

$$\rho_C =: 1 - \frac{I(X; Y)}{C} \quad (2.65)$$

- *channel efficiency*, η_C :

$$\eta_C =: \frac{I(X; Y)}{C} \quad (2.66)$$

Binary symmetric channel (BSC) capacity

A binary symmetric channel, BSC, is a channel for which the error probability p is the same for each symbol. As a result, the noise matrix has the following structure:

$$P_{\text{BSC}}(Y/X) = \begin{bmatrix} 1-p & p \\ p & 1-p \end{bmatrix} \quad (2.67)$$

The capacity of a BSC can be computed starting from (2.62); expressing $I(X; Y)$ by (2.58), we get:

$$C = \max_{P_i} [H(Y) - H(Y/X)] \quad (2.68)$$

Considering a binary source as input:

$$X : \begin{pmatrix} x_1 & x_2 \\ p_1 & p_2 \end{pmatrix}, \quad p_1 + p_2 = 1$$

we compute $H(Y/X)$ using (2.44), where the elements p_{ij} are calculated using (2.35):

$$P(XY) = P(X)P(Y/X) = \begin{bmatrix} p_1 & 0 \\ 0 & p_2 \end{bmatrix} \begin{bmatrix} 1-p & p \\ p & 1-p \end{bmatrix} = \begin{bmatrix} p_1(1-p) & p_1p \\ p_2p & p_2(1-p) \end{bmatrix} \quad (2.69)$$

and we obtain:

$$\begin{aligned} H(Y/X) &= \\ &= -[p_1(1-p)\log_2(1-p) + p_1p\log_2p + p_2p\log_2p + p_2(1-p)\log_2(1-p)] = \\ &= -[(1-p)\log_2(1-p) + p\log_2p](p_1 + p_2) = -p\log_2p - (1-p)\log_2(1-p) \end{aligned} \quad (2.70)$$

One may notice that, for a BSC, the average error, $H(Y/X)$, does not depend on the source, $P(X)$, but only on channel noise matrix $P(Y/X)$.

Replacing $H(Y/X)$ with (2.70) in (2.68) we get:

$$C_{\text{BSC}} = \max_{P_i} H(Y) + p \log_2 p + (1-p) \log_2 (1-p) \quad (2.68.a)$$

The maximum value for $H(Y)$ is obtained if all the received symbols have equal probabilities: $q_1=q_2=1/2$. These values are obtained as marginal probabilities from (2.69) as follows:

$$q_1 = p_1(1-p) + p_2 p = \frac{1}{2}$$

$$q_2 = p_1 p + p_2(1-p) = \frac{1}{2}$$

gives $p_1=p_2=1/2$. It means that the maximum value of the transformation is obtained when the symbols are used with equal probabilities and:

$$C_{\text{BSC}} = 1 + p \log_2 p + (1-p) \log_2 (1-p) \quad [\text{bits/symbol}] \quad (2.71)$$

Symmetric channel capacity

Symmetric channel is a channel that is symmetric at the input as well at the output. The noise matrix is as follows:

$$P(Y/X) = [q_{j/i}], \quad i = \overline{1, m}, \quad j = \overline{1, m}$$

where:

$$q_{j/i} = P(y_j/x_i) = \begin{cases} 1-p, & i = j \\ \frac{p}{m-1}, & i \neq j \end{cases} \quad (2.72)$$

Using (2.62) we have:

$$C_{\text{SC}} = \log_2 m + (1-p) \log_2 (1-p) + p \log_2 \frac{p}{m-1} \quad (2.73)$$

The reader is invited to demonstrate the relation.

Binary erasure channel (BEC) capacity

A BEC is a binary channel symmetric at the input, but asymmetric at the output.

The graph corresponding to such a channel is shown in Fig. 2.5. When y_3 is received, the input signal can be 0 or 1 with the same probability.

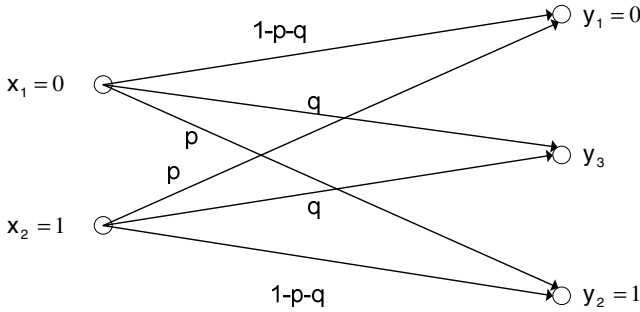


Fig. 2.5 The graph corresponding to a binary erasure channel

The noise matrix is:

$$P_{\text{BEC}}(Y/X) = \begin{bmatrix} 1-p-q & p & q \\ p & 1-p-q & q \end{bmatrix} \quad (2.74)$$

Using (2.62) the reader is invited to demonstrate:

$$C_{\text{BEC}} = (1-q)[1 - \log_2(1-q)] + (1-p-q)\log_2(1-p-q) + p\log_2 p \quad (2.75)$$

Remark

The errors that occur in memoryless channels are *independent errors*, caused by thermal noise. The formulae of error distribution are very complex and difficult to be computed; however, these formulae can be obtained by modelling the experimental results [1], [14].

For a BSC, with given p , the binomial law [9], [24] allows to calculate the probability of having t independent errors in an n length word:

$$P(t) = C_n^t p^t (1-p)^{n-t} \quad (2.76)$$

The probability of occurrence of t and less then t errors is given by:

$$P(e \leq t) = \sum_{e=0}^t p(e) = \sum_{e=0}^t C_n^e p^e (1-p)^{n-e} \quad (2.77)$$

Example 2.4

The symbols 0 and 1 are emitted at the input of a binary transmission channel. Statistical measurements show that, because of channel noise, both symbols are 10% erroneous, the process being time invariant. Knowing that the symbols 0 and 1 are transmitted in a ratio of 3/7, the transmission of a symbol is independent of the previously transmitted symbols and that 1000 symbols per second are being emitted (the duration of each symbol is the same), find:

- a) the statistical characterization of the transmission system
- b) the quantity of information obtained when 0 is emitted and the source average quantity of information

- c) source redundancy and efficiency
- d) rate of useful information transmitted through the channel
- e) channel efficiency.

Solution

The source and the receiver are modelled by the discrete r.v. X and Y , respectively, described by the following PMFs:

$$X : \begin{pmatrix} x_1 & x_2 \\ p_1 & p_2 \end{pmatrix}, \quad p_1 + p_2 = 1, \quad Y : \begin{pmatrix} y_1 & y_2 \\ q_1 & q_2 \end{pmatrix}, \quad q_1 + q_2 = 1$$

where x_1 and x_2 are the emission of 0 and 1 respectively, and p_1 and p_2 their corresponding emission probabilities. In the same way, y_1 and y_2 are the received 0 and 1 and q_1, q_2 the corresponding probabilities at receiver.

The modelling of the transmission channel is accomplished after computing the transition (noise) matrix $P(Y/X) = [q_{j/i}]$. The graphical representation of this transmission system is given in Fig 2.6.

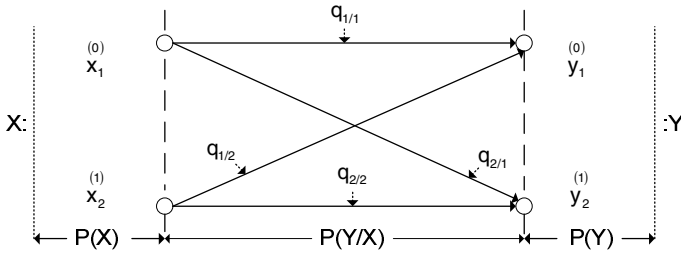


Fig. 2.6 Graphical representation of a binary transmission system

From the problem hypothesis we have:

$$\begin{cases} \frac{p_1}{p_2} = \frac{3}{7} \\ p_1 + p_2 = 1 \end{cases}$$

It follows: $P(X) = [0,3 \ 0,7]$

The given data of the problem indicate a transmission error of 10 % for both symbols ($q_{2/1} = q_{1/2} = 0.1$) and subsequently a correct transmission in 90% of the cases: $q_{1/1} = q_{2/2} = 0.9$. The transition (noise) matrix of the channel will be:

$$P(Y/X) = \begin{bmatrix} 0,9 & 0,1 \\ 0,1 & 0,9 \end{bmatrix}$$

Notice that $P(Y/X)$ is a stochastic matrix (the sum on each line is 1), which in practice means that if a symbol x_i is emitted, a symbol y_j is received, correct or erroneous. We can easily recognize the binary symmetric type of channel with $p=0,1$.

Knowing the source through $P(X)$ and the channel through $P(Y/X)$ we can easily calculate all the other PMFs.

Using (2.35) we obtain the joint probability matrix:

$$P(XY) = P(X)P(Y/X) = \begin{bmatrix} 0,3 & 0 \\ 0 & 0,7 \end{bmatrix} \begin{bmatrix} 0,9 & 0,1 \\ 0,1 & 0,9 \end{bmatrix} = \begin{bmatrix} 0,27 & 0,03 \\ 0,07 & 0,63 \end{bmatrix}$$

We invite the reader to check the calculus, calculating as marginal probabilities $P(X)$.

The received PMF can be calculated, according to (2.38), by summation on columns: $P(Y) = [0,34 \ 0,66]$.

Remark

$P(Y)$ can be calculated also using (2.32), as follows:

$$P(Y) = P(X)P(Y/X) = [0,3 \ 0,7] \cdot \begin{bmatrix} 0,9 & 0,1 \\ 0,1 & 0,9 \end{bmatrix} = [0,34 \ 0,66]$$

In order to find the matrix $P(X/Y)$, we use (2.33) to obtain p_{ij} elements. It results:

$$P(X/Y) = \begin{bmatrix} \frac{0,27}{0,34} & \frac{0,03}{0,66} \\ \frac{0,07}{0,34} & \frac{0,63}{0,66} \end{bmatrix}$$

It may be noticed that $P(X/Y)$, unlike $P(Y/X)$, is not a stochastic matrix. It was obtained through calculus and not experimentally, like $P(Y/X)$.

Using (2.11) (the definition of the self information), we have:

$$i(x_1) = -\log_2 p_1 = 1,737 \text{ bits}$$

The source average quantity of information defines its entropy and is computed using (2.12):

$$H(X) = -\sum_{i=1}^2 p_i \log_2 p_i = -(0,3 \log_2 0,3 + 0,7 \log_2 0,7) = 0,88 \text{ bits/symbol}$$

c) Using the definitions, we have:

- source absolute redundancy:

$$R_x = D(X) - H(X) = \text{ldm} - H(X) = 1 - 0,88 = 0,12 \text{ bits/symbol}$$

- source relative redundancy :

$$\rho_X = \frac{R_X}{D(X)} = 0,12 \text{ or } 12\%$$

- source efficiency:

$$\eta_x = \frac{H(X)}{D(X)} = 0,88 \text{ or } 88\%$$

d) The transinformation rate through the channel is:

$$\dot{I}(X; Y) = \dot{M} I(X; Y)$$

where $\dot{M} = 10^3$ symbols/second = 10^3 Bd.

The transinformation can be calculated using (2.53) or the relationships between entropies, the last one being used as follows:

$$I(X; Y) = H(Y) - H(Y/X)$$

The binary channel is symmetric, so we have the following equations:

$$H(Y/X) = -p \log p - (1-p) \log(1-p),$$

hence

$$I(X; Y) = -(0,34 \log 0,34 + 0,66 \log 0,66) + 0,11 \log 0,11 + 0,91 \log 0,9 = 0,456 \text{ bits/symbol}$$

$$\dot{I}(X; Y) = 10^3 \cdot 0,456 = 456 \text{ bits/sec}$$

e)

$$\eta_C = \frac{I(X; Y)}{C} = \frac{0,456}{1 - 0,469} = \frac{0,456}{0,531} \approx 0,86 \text{ or } 86\%$$

The maximum efficiency $\eta_C = 1$, corresponding to a transformation that equals the channel capacity, can be obtained only when the source X is equally probable, which means (for the given system) that a source processing (encoding) must be performed before the transmission.

Example 2.5

Two binary symmetric channels given by $p_1 = 10^{-1}$ and $p_2 = 10^{-2}$ are cascaded. Find the:

- equivalent noise matrix of the cascade
- equivalent channel capacity

Solution

a) The noise matrices of two BSC are:

$$P(Y/X) = \begin{bmatrix} 1-p_1 & p_1 \\ p_1 & 1-p_1 \end{bmatrix}, \quad P(W/Z) = \begin{bmatrix} 1-p_2 & p_2 \\ p_2 & 1-p_2 \end{bmatrix}$$

The equivalent noise matrix of the cascade is

$$\begin{aligned} P(W/X) &= P(Y/X) \cdot P(W/Z) = \begin{bmatrix} 1-p_1 & p_1 \\ p_1 & 1-p_1 \end{bmatrix} \cdot \begin{bmatrix} 1-p_2 & p_2 \\ p_2 & 1-p_2 \end{bmatrix} = \\ &= \begin{bmatrix} (1-p_1) \cdot (1-p_2) + p_1 p_2 & p_2(1-p_1) + p_1(1-p_2) \\ p_1(1-p_2) + p_2(1-p_1) & p_1 p_2 + (1-p_1) \cdot (1-p_2) \end{bmatrix} \end{aligned}$$

It follows that:

$$\begin{cases} q_{1/1} = q_{2/2} = p_1 p_2 + (1-p_1) \cdot (1-p_2) = 1 - (p_1 + p_2) + 2p_1 p_2 \\ q_{1/2} = q_{2/1} = p_1 + p_2 - 2p_1 p_2 \end{cases}$$

meaning that the equivalent channel is BSC too, with

$$p_{\text{ech}} = q_{2/1} = q_{1/2} = p_1 + p_2 - 2p_1 p_2$$

The quality of the equivalent channel is worst than the worst channels in the cascade:

$$p_{\text{ech}} = p_1 + p_2 - 2p_1 p_2 \approx p_1 + p_2 = 10^{-1} + 10^{-2}$$

b)

$$\begin{aligned} C_{\text{ech}} &= 1 + p_{\text{ech}} \log_2 p_{\text{ech}} + (1 - p_{\text{ech}}) \log_2 (1 - p_{\text{ech}}) = \\ &= 1 + 0,11 \log_2 0,11 + 0,89 \log_2 0,89 = \\ &= 1 - 0,3503 - 0,1996 = 0,4974 \text{ bits/binary symbol} \end{aligned}$$

2.8.5 Shannon Capacity

Capacity of a transmission channel, modelled by $P(Y/X)$ – the noise matrix, was defined in 2.8.4. In most practical cases the noise matrix is not known, the channel being specified by some parameters which can be measured experimentally much easier, like the bandwidth (B) and the signal/noise ratio (SNR - ξ).

Theoretically it is possible to transmit any quantity of information on a channel. The maximum information rate that can be transmitted in real time is limited, and this limit defines the *channel capacity*. This limitation is determined by channel characteristics and it stands in both digital and analogue transmissions. We will compute channel capacity (without trying to impose it as a demonstration) for digital systems, as they are the most used [6]; as a matter of fact, the analogue information can be considered a limit case of the digital information ($m \rightarrow \infty$) [32], [9].

At channel input we consider a discrete source described by:

- decision rate: D [bits/s], therefore the source is assumed to be equally probable (in practice this is obtained after encoding)
- moment rate: M [Bd], which shows how fast the carrier signal varies.
- source alphabet, formed by m specific states of a moment (these can be levels, frequencies or phases).

As previously shown in 2.7, these three parameters are linked by:

$$\dot{D} = \dot{M} \log_2 m \quad (2.27)$$

In real cases the receiver should be able to distinguish, in the presence of noise, two successive moments for which the characteristic parameter takes, in the worst case, two consecutive values from the m possible.

In order to be able to transmit a decision rate \dot{D} , the channel must provide:

- a *time resolution*, meaning it should allow the signals characteristic parameter to vary from one moment to another, or during one moment.
- an *amplitude resolution*, such as the m possible values of the characteristic parameter may be distinguished even in the presence of noise.

Figure 2.7 shows a graphical illustration of these two requirements:

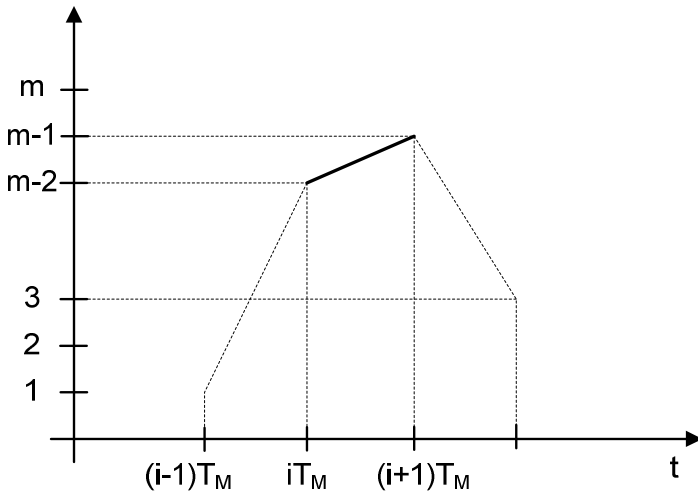


Fig. 2.7 Illustration of time and amplitude resolutions

Time resolution

Any real transmission channel contains reactance that oppose to fast signal variations, leading to an inertial behaviour of the channel. This phenomenon exists both in frequency and in time.

In frequency: channel attenuation is a function of frequency, channel behaving as low-pass filter (LPF).

In time: channel unit-step signal response has a finite slope defined by the rise time (t_r) [15], [27]. When talking about real and ideal channels, the following empirical relation stands between t_r and B :

$$Bt_c \cong 0,35 \div 0,45 \quad (2.78)$$

which shows that for a channel of bandwidth B , the parameters cannot vary at any speed, being limited by t_r . Therefore the moment duration $T_M = 1/\dot{M}$ is also limited by t_r :

$$T_M \sim t_r \Rightarrow \frac{1}{\dot{M}} \sim \frac{0,4}{B} \Rightarrow \dot{M} \sim B \quad (2.79)$$

In 1928, H. Nyquist has proved the required relation between \dot{M} and B in order to achieve a time resolution (no inter-symbol interference)

$$\dot{M}_{\max} = 2B, \text{ for an ideal filter,} \quad (2.80)$$

known in literature as *the Nyquist theorem* [20], [23].

For real channels, which are not ideal low-pass filters we have:

$$T_M \cong 2t_r \quad (2.81)$$

From (2.78) we get $t_c = 0.4/B$ and (2.81) becomes:

$$T_M = \frac{0.8}{B}, \quad \text{and for real channels, we have:}$$

$$\dot{M}_{\max} = 1.25 \cdot B \quad (2.80a)$$

Amplitude resolution

Besides channel inertia that gives its low-pass filter behaviour, the noise on the channel added to the transmitted symbols, deteriorates the detection of the m values corresponding to a moment.

The signal power P_S being limited, it is impossible to recognize an infinity of different values of m in the presence of noise (the noise power is P_N).

In 1948, Shannon demonstrated that the theoretical limit for m in the presence of additive white Gaussian noise (AWGN), is:

$$m_{\max} = \sqrt{\frac{P_S + P_N}{P_N}} = \sqrt{1 + \xi} \quad (2.82)$$

where $\xi = P_S/P_N$ is the signal/noise (SNR).

Channel capacity

Channel capacity is defined as the maximum decision rate that can be error free transmitted through that channel, assumed an ideal Low Pass Filter (LPF) with bandwidth B :

$$C =: \dot{D}_{\max} = \dot{M}_{\max} \log_2 m_{\max} = 2B \cdot \log_2(\sqrt{1 + \xi}) = B \cdot \log_2(1 + \xi) \quad (2.83)$$

and is known as *Shannon capacity formula* (Shannon 3rd theorem).

Remark

If $SNR(\xi)$ is given in dB:

$$\xi_{[dB]} = 10 \lg \xi$$

the relation (2.83) becomes:

$$C \cong \frac{1}{3} B \xi_{[dB]} \tag{2.83.a}$$

Shannon capacity formula (2.83), given by Shannon in 1948, shows the theoretical limit of the maximum information rate through the channel in error free transmission.

Despite the relation (2.83) is a theoretical limit, impossible to be reached in real transmissions, it is remarkably useful in applications, allowing a comparison and an evaluation of different transmission systems.

Fig. 2.8 shows the graphical representation of this equation.

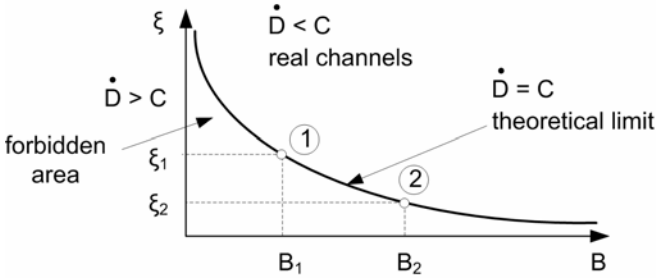


Fig. 2.8 Graphical representation of channel capacity

Interpretations of Shannon capacity formula

1) $C \sim B$ and $C \sim \log_2(1 + \xi)$, which means that reducing the bandwidth and keeping C constant, ξ must be seriously improved, given the logarithmic dependence between C and ξ .

2) Given the proportionality between capacity and bandwidth, the following question arises: is it possible to increase the capacity to infinity based on the increasing of bandwidth? The answer is negative, the reason being obvious: increasing the bandwidth, the noise power P_N , increases too, causing a decrease of $SNR(\xi)$, for constant P_S . The proof is immediate: the calculations is made under the assumption of AWGN with spectral density power N_0 constant:

$$C_\infty = \lim_{B \rightarrow \infty} B \log_2 \left(1 + \frac{P_S}{BN_0} \right) = \frac{P_S}{N_0} \ln 2 = ct \tag{2.84}$$

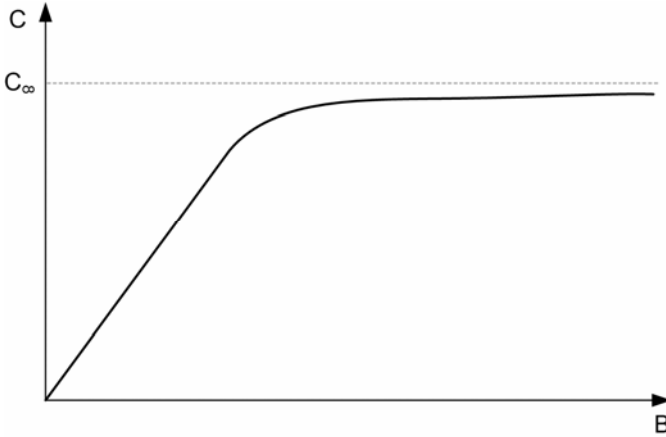


Fig. 2.9 Bandwidth - capacity dependency

Therefore, increasing the capacity beyond a certain limit is not rational, the corresponding capacity gain being very low (Fig 2.9).

3) The formula (2.83), under the assumption of AWGN can be written as:

$$C = B \cdot \log_2 \left(1 + \frac{P_S}{BN_0} \right)$$

If the power of the signal is expressed as:

$$P_S = \frac{E_b}{T_M}$$

where E_b is the average value of the bit energy and we assume

- binary transmission ($m=2$): $\Rightarrow \dot{D} = \dot{M}$
- transmission at capacity (ideal transmission) : $\Rightarrow \dot{D} = C$, the former formula becomes :

$$\frac{C}{B} = \log_2 \left(1 + \frac{C}{B} \cdot \frac{E_b}{N_0} \right) \text{ or } \frac{E_b}{N_0} = \frac{2^{\frac{C}{B}} - 1}{\frac{C}{B}}$$

At limit $B \rightarrow \infty$, it gives :

$$\lim_{B \rightarrow \infty} \frac{E_b}{N_0} = \lim_{B \rightarrow \infty} \frac{2^{\frac{C}{B}} - 1}{\frac{C}{B}} = \ln 2 = 0,693 .$$

If expressed in dB, we obtain :

$$\lim_{B \rightarrow \infty} \frac{E_b}{N_0} = -1,59\text{dB} \cong -1,6\text{dB} \quad (2.85)$$

known as *Shannon limit for an AWGN* and gives the minimum required $\frac{E_b}{N_0}$ ratio for error-free transmission

The ratio:

$$\eta = \frac{\dot{D}}{B} \text{ [bits/sec/Hz]} \tag{2.86}$$

is defined as *spectral efficiency (bandwidth efficiency)* [34].

Interpretations of Shannon limit

- it is a theoretical limit ($\dot{D} = C$), showing that for infinite bandwidth ($B \rightarrow \infty$), the ratio $\frac{E_b}{N_0}$ approaches the limit of -1,6 dB (in this case $C = C_\infty = \frac{P_S}{N_0} \ln 2$)
- the *capacity boundary* [9], defined by the curve for the critical bit rate $\dot{D} = C$, separates combinations of system parameters that have the possibility for supporting real-time error-free transmissions ($\dot{D} < C$), from those for which real-time error-free transmissions are impossible ($\dot{D} > C$).

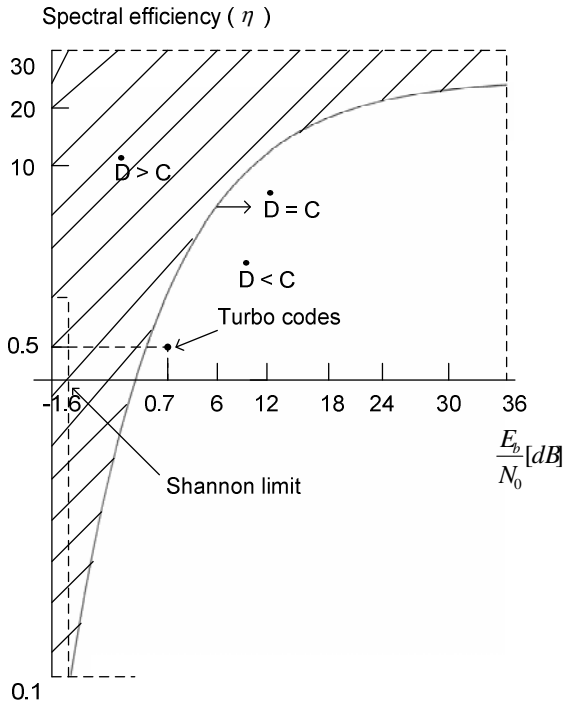


Fig. 2.10 Bandwidth-efficiency diagram

Remark

Turbo codes, invented in 1993 [3], with iterative decoding have almost closed the gap between capacity limit and real code performances. They get BER=10⁻⁵ at $\frac{E_b}{N_0} = 0,7 \text{ dB}$ with spectral efficiency of 0,5 bit/second/Hz [34].

4) The same capacity can be obtained using different values for ξ and B: we can use a narrow bandwidth B₁ and a channel with a very good SNR ξ_1 (this case corresponds to system 1 shown in Fig. 2.8); a noisy channel with a small ξ_2 requires a wider bandwidth B₂ (system 2, Fig. 2.8) to provide the same capacity.

5) Relation (2.83) gives a theoretical limit of the maximum transmissible decision rate. On real channels: $\dot{D}_{\max \text{ real}} < C$. This limit is not obtained automatically; it can be obtained by processing the source before transmission, that is, by matching the source to the channel (described by B and ξ) through coding and modulation.

A suggestive graphical representation of the relations between the information source and the channel is provided in Fig. 2.11 [6].

If $\dot{D} > C$, a real time transmission is no longer possible; in this case the same decision quantity $D = \dot{D} \cdot T$ can be transmitted using an initial processing; the decision quantity D is stored in a memory and then transmitted through a compatible channel ($\dot{D} \leq C$). It is obvious that the transmission time T increases, the transmission not being performed in real time. In practice this situation occurs when images are transmitted by space probes and the channel capacity is much smaller compared to the source information rate.

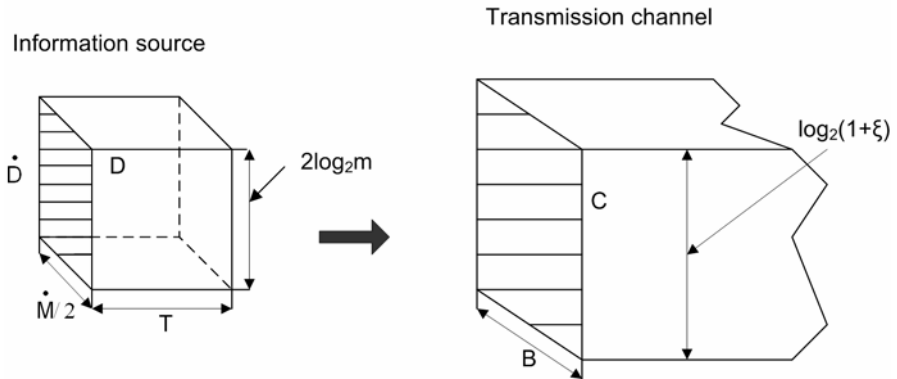
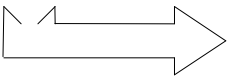


Fig. 2.11 Graphical representation of the relations between the information source and the channel

Table 2.2 Illustration of time and amplitude resolution requirements

Time resolution	Moment rate $\dot{M} = \frac{1}{T_M}$	$\dot{M} \leq 2B$	Bandwidth B
Amplitude resolution	m values/moment $m \leq \sqrt{1 + \xi}$		Signal/noise ratio $\xi = \frac{P_S}{P_N}$
	Decision rate $\dot{D} = M \log_2 m$	$\dot{D} \leq C$	Capacity $C = B \log_2(1 + \xi)$

Example 2.6 [21]

A black-and-white still image (a picture) is decomposed into $n = 4.8 \cdot 10^5$ elements (pixels). Experiments have shown that if 100 levels quantify the brightness intensity of each pixel, the reconstructed image seems natural. The 100 levels of grey are assumed to be equally probable. Find the:

- a) average quantity of information supplied by one pixel and that of the whole image.
- b) minimum value of the required signal/noise ratio, and the transmission time for one picture on a voice channel with bandwidth (300 - 3400)Hz .
- c) transmission time compared to b) if the transmission line has the same ξ but a double bandwidth.
- d) minimum bandwidth necessary for transmitting a dynamic image (video),

knowing that 25 images/s (\dot{M}) are needed by the human visual system in order to obtain a moving image sensation; is it possible such a transmission on the voice channel used at point b)?

Solution

a) the information corresponding to one pixel is contained in its brightness; the 100 levels being equally probable, the average information quantity for a pixel is maximum and using (2.13):

$$D_{\text{pixel}} = \log_2 m = \log_2 100 = 6,64 \text{ bits/element (pixel)}$$

The average quantity of information of the entire image is:

$$D_{\text{image}} = n D_{\text{pixel}} = 4,8 \cdot 10^5 \cdot 6,64 = 3,2 \text{ Mb}$$

b) for the amplitude resolution, from (2.82) we have:

$$m = \sqrt{1 + \xi} \Rightarrow \xi \cong 10^4, \text{ or in dB: } \xi_{\text{dB}} = 10 \lg \xi = 40 \text{dB}$$

Assuming that the transmission is made at channel full capacity, from (2.83a) we get:

$$C_b = \dot{D} = \frac{D}{T_b} = \frac{1}{3} B_b \xi_{[dB]}$$

where T_b represents the transmission time and $B_b = 3400 - 300 = 3,1 \cdot 10^3$ Hz the bandwidth of the telephone line. From (2.83a) we obtain:

$$T_b = \frac{3D}{B_b \xi_{[dB]}} = \frac{3 \cdot 3,2 \cdot 10^6}{3,1 \cdot 10^3 \cdot 40} \cong 77s$$

c) from (2.83), replacing B with 2B, we get:

$$T_c = T_b / 2 = 38,5 s.$$

d) decision rate corresponding to a dynamic image (video) is:

$$\dot{D}_d = \dot{M} \cdot \dot{D} = 25 \cdot 3,2 \cdot 10^6 = 80Mb/s$$

Assuming that the channel is used at full capacity, this rate becomes:

$$\dot{D}_d = C_d = \frac{1}{3} B_d \xi_{[dB]}$$

giving:

$$B_d = \frac{3\dot{D}_d}{\xi_{[dB]}} = \frac{3 \cdot 80 \cdot 10^6}{40} = 6MHz$$

Capacity of the telephone channel from point b) is:

$$C_b = \frac{1}{3} B_b \xi_{[dB]} = \frac{1}{3} \cdot 3,1 \cdot 10^3 \cdot 40 \cong 41kb/s$$

whereas:

$$\dot{D}_d = 80Mb/s > C_b = 41kb/s,$$

therefore, the real-time transmission is not possible.

2.9 Memory Sources (Markov Sources)

Taking into account the definition given in 2.1 we may say that most of the real information sources such as voice signals, TV signals a. s. o. are not memoryless sources. For all these sources the emission of a symbol depends on one or more previously emitted symbols.

A source is an *m-order memory source*, if the generation of the (m+1) order symbol depends on the previously m generated symbols.

The mathematical theory of memory sources is Markov chains theory [11], [20], [23], [24].

2.9.1 *Finite and Homogeneous Markov Chains*

Only few mathematical concepts share potentialities comparable to those offered by the concept of markovian dependency. Some fields where this theory has been applied are listed below:

- biology
- medicine
- demography
- meteorology
- economy
- reliability studies
- research on pollution
- marketing
- financial transactions
- pattern recognition theories
- natural language modelling

Brief historical overview

The concept of markovian dependency appears explicitly for the first time in 1906 in one of A. Markov papers, a Russian mathematician. He studied sequences of dependent variables that are named, in his memory, Markov chains. Markov himself studied the succession of vowels and consonants in Pushkin's novel "Evgeni Onegin" and reached the conclusion that this succession may be seen as a two state, homogenous dependent chain.

Nowadays there are an enormous number of papers that deal with Markov chains. It is necessary to emphasize the significant contribution brought to the theory of finite Markov chains and its generalization by the godfathers of the Romanian school of statistics and probability: academicians Octav Onicescu and Gheorghe Mihoc. They initiated the research of generalizing the concepts of markovian dependency.

Be a system (source) S with a finite alphabet $S = \{s_1, s_2, \dots, s_M\}$. The generation of a symbol is dependent of the m previous symbols. This source can be modelled by a dependent random variable $X(n)$, with values in S , where n indicates the time moment. Time evolution of the source is known statistically as:

$$P(X(0) = s_i, \dots, X(i) = s_j, \dots, X(n) = s_k)$$

We call a *finite and homogenous Markov chain* a sequence of dependent r.v. $X(n)$ with the following properties:

$$\begin{aligned} P(X(n) = s_j / X(n-1) = s_i, X(n-2) = s_k, \dots, X(1) = s_p) \\ = P(X(n) = s_j / X(n-1) = s_i) = \end{aligned} \tag{2.87.a}$$

$$= P(s_j / s_i) = p_{j/i}, \forall n \tag{2.87.b}$$

Relation (2.87.a) defines the *Markov property*, revealing the one-step memory, meaning that the entire source history is contained in its most recent memory.

Relation (2.87.b) indicates the *homogeneity*, that is, transitions from one state to another are not time dependent.

All conditional probabilities $p(s_j/s_i) = p_{j/i}$ generate the transition probability matrix (Markov matrix) as:

$$P(s_j/s_i) = M = \begin{bmatrix} P_{1/1} \cdots P_{j/1} \cdots P_{M/1} \\ \cdots \cdots P_{j/i} \cdots \cdots \\ P_{1/M} \cdots P_{j/M} \cdots P_{M/M} \end{bmatrix} \quad (2.88)$$

Markov matrix is a stochastic, squared matrix:

$$\sum_{j=1}^M p_{j/i} = 1, \quad \forall i = \overline{1, M} \quad (2.89)$$

For a Markov source we may also know the initial PMF (probabilities at zero moment):

$$P^{(0)} = [p_i^{(0)}], \quad i = \overline{1, M}, \quad \text{and} \quad \sum_{i=1}^M p_i = 1 \quad (2.90)$$

A Markov chain can be illustrated by a graph made up of nodes, that represent the states, and arrows, that connect the nodes, representing transition probabilities.

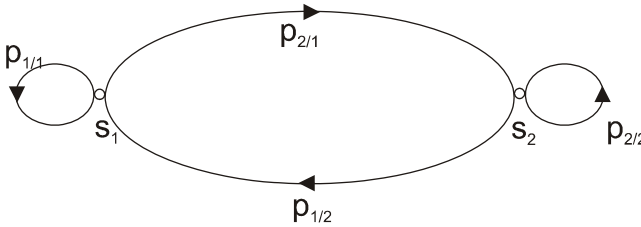


Fig. 2.12 The graph corresponding to a Markov chain with two states

Transition probabilities

The probability of transitioning from the state s_i to the state s_j in m steps is given by

$$p_{j/i}^{(m)} = p(X(n+m) = s_j / X(n) = s_i), \quad \forall n \quad (2.91)$$

One may notice that the homogeneity also extends to the m -step transition.

Let us consider two states s_i and s_j that can be reached in two steps $m=2$ on the links shown in Fig. 2.13.

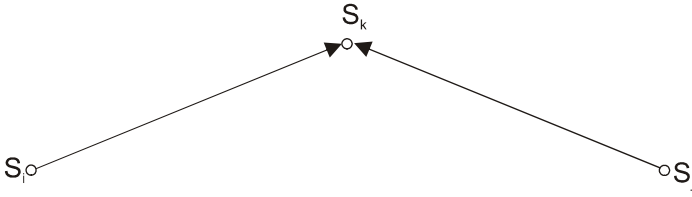


Fig. 2.13 Transition from s_i to s_j in two steps

For the way shown in Fig. 2.13, the transition probability from s_i to s_j in two steps is given by:

$$p_{j/i}^{(2)} = p_{k/i} \cdot p_{j/k} \quad (2.92)$$

The transition probability from s_i to s_j in two steps on any way is given by:

$$p_{j/i}^{(2)} = \sum_{k=1}^M p_{k/i} \cdot p_{j/k} \quad (2.93.a)$$

We must notice that $p_{j/i}^{(2)}$ are the elements of the second order power of the transition matrix M :

$$P^{(2)} = M^2 \quad (2.93.b)$$

By induction it can be demonstrated that:

$$p_{j/i}^{(n+m)} = \sum_{k=1}^M p_{k/i}^{(n)} \cdot p_{j/k}^{(m)}, \quad \forall n, m \in \mathbb{N}, \quad \forall i, j \in S \quad (2.94.a)$$

Remark

These equations are known as *Chapman-Kolmogorov relations* and can be written as follows:

$$P^{(n+m)} = M^n \cdot M^m \quad (2.94.b)$$

Markov chain evolution

Given a Markov chain by the initial PMF, $P^{(0)}$ and the transition matrix M , the probability of the system to pass to a certain state in n steps starting from the initial state is:

$$P^{(n)} = P^{(0)} \cdot M^n \quad (2.95)$$

Markov chains classification

A Markov chain is called *regular* if there is an $n_0 \in \mathbb{N}$ (natural set) such that M^{n_0} is a regular matrix (all its elements are strictly positive).

A Markov chain is called *stationary* if the n-step transition probabilities converge, when $n \rightarrow \infty$, towards limits independent of the initial state.

$$\lim_{n \rightarrow \infty} p_{j/i}^{(n)} = p_j^* \tag{2.96}$$

p_j^* indicates that the starting state is of no importance. The *stationary state PMF* is:

$$P^* = [p_j^*], \quad j = \overline{1, M}; \quad \sum_{j=1}^M p_j^* = 1 \tag{2.97}$$

and represents

$$P^{(n)} = M^n = \begin{bmatrix} P^* \\ \vdots \\ P^* \end{bmatrix} = \Pi \tag{2.98}$$

Relation (2.98) indicates that we obtained a matrix having all the rows identical with the PMF of the stationary state.

We compute this stationary state by solving the following system:

$$\begin{cases} P_M^* M = P^* \\ \sum_{j=1}^M p_j^* = 1 \end{cases} \tag{2.99}$$

In [11], [24] is demonstrated that:

- for a stationary Markov chain, the stationary state PMF is unique.
- a regular Markov chain accepts a stationary state.

A Markov chain is called *absorbent* if it has at least an absorbent state for which $p_{i/i} = 1$. In [11] is demonstrated that an *absorbent chain* is not regular.

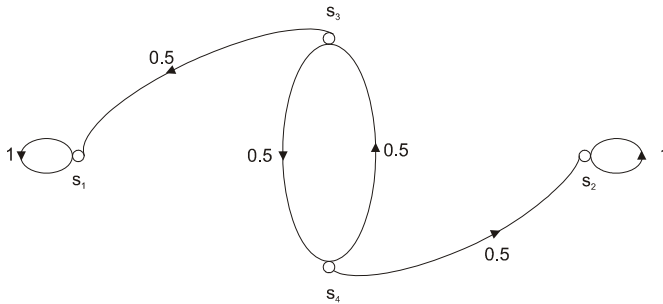


Fig. 2.14 Absorbent Markov chain

The transition matrix for the chain shown in Fig. 2.14 is:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0,5 & 0 & 0 & 0,5 \\ 0 & 0,5 & 0,5 & 0 \end{bmatrix}$$

We must notice that s_1 and s_2 are absorbent states meaning that if the system reaches one of these states it “hangs” on to it.

Example 2.7

One classical example of using Markov chains is weather forecasting. In meteorological stations the weather is observed and classified daily at the same hour, as follows:

- $s_1 = S$ (sun)
- $s_2 = C$ (clouds)
- $s_3 = R$ (rain)

Based on daily measurements, meteorologists computed the transition probability matrix:

$$M = \begin{bmatrix} 0,6 & 0,2 & 0,2 \\ 0,3 & 0,4 & 0,3 \\ 0,2 & 0,2 & 0,6 \end{bmatrix}$$

Some questions raised:

- knowing that today is a sunny day, what is the probability that the forecast for the next seven days is: $s_1 s_1 s_2 s_1 s_2 s_3 s_1$?

The answer is:

$$\begin{aligned} & p(s_1) \cdot p(s_1/s_1) \cdot p(s_1/s_1) \cdot p(s_2/s_1) \cdot p(s_1/s_2) \cdot p(s_2/s_1) \cdot p(s_3/s_2) \cdot p(s_1/s_3) = \\ & = 1 \cdot 0,6 \cdot 0,6 \cdot 0,2 \cdot 0,3 \cdot 0,2 \cdot 0,3 \cdot 0,2 = 2,592 \cdot 10^{-4} \end{aligned}$$

- what is the probability that the weather becomes and remains sunny for three days, knowing that the previous day it had rained?

$$\begin{aligned} & p(s_3) \cdot p(s_1/s_3) \cdot p(s_1/s_1) \cdot p(s_1/s_1) \cdot [1 - p(s_1/s_1)] = \\ & = 1 \cdot 0,2 \cdot 0,6 \cdot 0,6 \cdot (1 - 0,6) = 2,88 \cdot 10^{-2} \end{aligned}$$

2.9.2 Entropy of m -th Order Markov Source

Consider a source having a finite alphabet: $A = \{a_1 \dots a_M\}$ and assume that the occurrence of the $(m+1)$ th symbol depends on the previously m : $a_j/a_{i_1}a_{i_2} \dots a_{i_m}$.

We denote the sequence $a_{i_1}a_{i_2} \dots a_{i_m}$ by s_i ($a_{i_1}a_{i_2} \dots a_{i_m} = s_i$). The total number of different sequences that can be created using an M size alphabet is M^m , so $i = 1, M^m$.

The quantity of information obtained at the emission of a_j conditioned by the sequence s_i is:

$$i(a_j/s_i) = -\log_2 p(a_j/s_i) \quad (2.100)$$

The average quantity of information obtained at the emission of any a_j conditioned by s_i will be the conditional entropy:

$$H(A/s_i) = -\sum_{j=1}^M p(a_j/s_i) \log_2 p(a_j/s_i) \quad (2.101)$$

The average quantity of information obtained at the emission of any symbol a_j conditioned by any sequence s_i of m -symbols, represents the *entropy of m -step memory source*

$$\begin{aligned} H_m(A) &= \sum_{i=1}^{M^m} p(s_i) H(A/s_i) = -\sum_{i=1}^{M^m} \sum_{j=1}^M p(s_i) p(a_j/s_i) \log_2 p(a_j/s_i) = \\ &= -\sum_{i=1}^{M^m} \sum_{j=1}^M p(s_{ji}) \log_2 p(a_j/s_i) \end{aligned} \quad (2.102)$$

where $s_{ji} = \frac{a_{i1} \dots a_{im} a_j}{s_i}$.

Remarks

- In the particular case $p(a_j/s_i) = p(a_j)$ corresponding to a memoryless source, we obtain (2.12) relation corresponding to a memoryless discrete source (MDS).
- $H_m(A)$ is always smaller compared to the one corresponding to the same source, but memoryless: the average non-determination quantity per symbol a_i decreases because of the symbol correlation.

Example 2.8

Find the corresponding graph for a two-step binary source (second order memory source) and calculate its entropy if $p(s_i)$ are the ones corresponding to the stationary state and the elements $p(a_j/s_i)$ are chosen randomly.

Solution

Consider the binary ($M=2$) source having the alphabet:

$$A = \{0, 1\}$$

The $m = 2$ -step memory source will have $M^m = 4$ states:

$$s_1 = 00, s_2 = 01, s_3 = 10, s_4 = 11.$$

The graph corresponding to the source can be found taking into consideration that at the emission of symbol a_j , from a state s_i only certain states can be reached. (Fig 2.15)

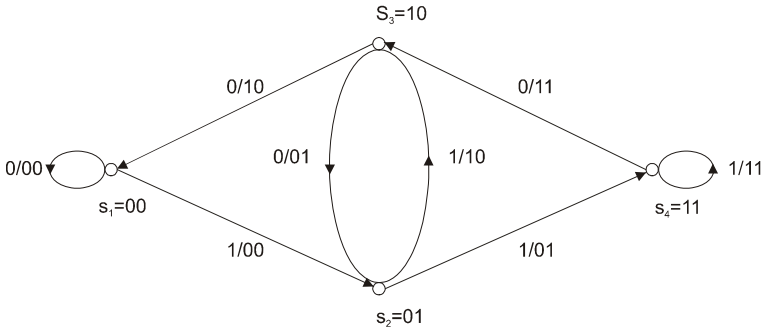


Fig. 2.15 Graph corresponding to a 2-step memory binary source

We choose the transition matrix of this source as:

$$M = \begin{bmatrix} 1/4 & 3/4 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 \end{bmatrix}$$

Notice that the matrix M contains zero elements, therefore in order to calculate the stationary distribution, we should first be certain it exists. Based on the theorems stated in 2.9.1, we verify if the source is regular. For $n_0=2$, M^2 has all the elements strictly positive, so the source is regular, therefore it accepts a stationary state which can be established by solving the system (2.49).

$$\begin{cases} [p_1^* & p_2^* & p_3^* & p_4^*] M = [p_1^* & p_2^* & p_3^* & p_4^*] \\ p_1^* + p_2^* + p_3^* + p_4^* = 1 \end{cases}$$

We obtain:

$$P^* = \begin{bmatrix} 2/11 & 3/11 & 3/11 & 3/11 \end{bmatrix}$$

The elements of this source, necessary for entropy computation are contained in the table:

s_i			$p(s_i) = p_i^*$	$p(a_i/s_i)$
a_{i1}	a_{i2}	a_i		
0	0	0	2/11	1/4
0	0	1	2/11	3/4
0	1	0	3/11	1/2
0	1	1	3/11	1/2
1	0	0	3/11	1/2
1	0	1	3/11	1/2
1	1	0	3/11	1/2
1	1	1	3/11	1/2

Replacing in (2.52) we find: $H_2(A) = 0,549$ bits/binary symbol.

Example 2.9

A computer game implies the use of only two keys (a_1, a_2). If the pressing of a key depends on the key pressed before, a Markov chain may model the game, its transition matrix being:

$$M = \begin{bmatrix} 1/3 & 2/3 \\ 3/4 & 1/4 \end{bmatrix}$$

a) Which is the average quantity of information obtained at one key pressing? Assume that $p(a_1)$ and $p(a_2)$ are the stationary state probabilities. Compare this value with the one obtained for a memoryless source model.

b) Draw the graph corresponding to the 2-nd order Markov source model. How will be the average quantity of information obtained at a key pressing compared to that from a)?

Solution

The stationary state PMF can be established by solving the system:

$$\begin{cases} \begin{bmatrix} p_1^* & p_2^* \end{bmatrix} M = \begin{bmatrix} p_1^* & p_2^* \end{bmatrix} \\ p_1^* + p_2^* = 1 \end{cases}$$

We obtain: $P^* = [9/17 \ 8/17]$. The average quantity of information obtained for a key pressing represents the first order Markov entropy of the source. Using (2.102) we obtain:

$$\begin{aligned} H_1(S) &= -\sum_{i=1}^2 \sum_{j=1}^2 p(s_i) p(a_j/s_i) \log_2 p(a_j/s_i) = \\ &= -\left(\frac{9}{17} \cdot \frac{1}{3} \cdot \log_2 \frac{1}{3} + \frac{9}{17} \cdot \frac{2}{3} \cdot \log_2 \frac{2}{3} + \frac{8}{17} \cdot \frac{3}{4} \cdot \log_2 \frac{3}{4} + \frac{8}{17} \cdot \frac{1}{4} \cdot \log_2 \frac{1}{4} \right) = \\ &= 0.867 \text{ bits/symbol} \end{aligned}$$

In the case of a memoryless source modelling of the game, we have:

$$H(S) = -\sum_{i=1}^2 p_i \log_2 p_i = -\frac{9}{17} \cdot \log_2 \frac{9}{17} - \frac{8}{17} \cdot \log_2 \frac{8}{17} = 0.9974 \text{ bits/symbol}$$

so $H_1(S) < H(S)$, as expected.

A two step memory binary source is modelled as shown in Fig. 2.14. The average quantity of information obtained at a key pressing will be the entropy of second order Markov source and will be smaller than the one corresponding to the first order Markov source model.

2.9.3 Applications

From the huge number of Markov chains applications we will choose some from the artificial intelligence and pattern recognition fields.

Natural language modelling [5]

Any natural language has an alphabet, a dictionary and a syntax. Take for instance the Latin alphabet: it contains 26 letters and a space between words, so a total of 27 symbols. The simplest model associated to such a language is the one corresponding to a discrete memoryless source (the zero order approximation) in which all symbols are equally probable. The average quantity of information corresponding to a symbol will be:

$$\dot{D}(S) = \text{ld}27 = 4,75\text{bits/symbol}.$$

A sequence of symbols emitted by such a model will not mirror the structure of a certain language (French, English, Italian, Romanian etc).

Considering the frequencies of the alphabet letters for a certain language, a slight reflection of the language can be obtained. It means the first order approximation, which leads to the average quantity of information per letter (for French [5]).

$$\dot{D}(S) = \text{ld}27 = 4,75\text{bits/symbol}$$

A better approximation of this information source is made with a first order Markov source in which case the joint probabilities $s_{ij} = a_i a_j$ and the conditioned probabilities $p(a_j/a_i)$ must be known. In this case, in a letter sequence, we may recognize with certainty the language type.

Better models of a natural language can be achieved using second order Markov sources. If there are constraints regarding the word length, if the symbols are replaced by words and the alphabet by the language dictionary, and if the syntax is taken into consideration, we can find complex sentences in that language, with no special signification.

Acting this way and imposing even more strict constraints, poems can be elaborated on the computer, or literary works can be authenticated etc.

The more constraints, the less system uncertainty and the number of expressible messages is reduced. The constraint degree in a system (language) can be estimated by its redundancy R:

$$R = \frac{\log_2 M_p - \log_2 M_u}{\log_2 M_u} = \frac{I_i}{\log_2 M_u} \quad (2.103)$$

where M_p represents the number of possible messages without constraints and M_u the number of useful messages taking into consideration the constraints. Internal information of the system is defined (I_i)

$$I_i = \log_2 M_p - \log_2 M_u. \quad (2.104)$$

In [5] it is shown that, for the French language, the average number of letters per word being 5, the number of 5-letter messages that can be achieved using 26 symbols is: $M_p = 26^5 \approx 12 \cdot 10^6$. The French dictionary contains approximately 7000 5-letter words, so $M_u=7000$, from which using (2.103) and (2.104) results

that the French language redundancy is $\approx 45\%$ and the internal information is 2,15 bits/letter.

Voice modelling

The voice average rate [6] is 80...200 words/minute. For 5-letter average length words, in a zero order approximation memoryless, results $D \cong 4,7$ bits/letter

therefore a rate $\dot{D} \cong 40 \dots 100$ bits/s. The real rate for a better approximation (superior order Markov sources), in which $H(S) \cong 1,5$ bits/letter, is much lower:

$\dot{H}(S) = 12 \dots 30$ bits/s. It is necessary to emphasize that this value represents only the semantic rate of the voice (the literal signification of the words). In reality the voice signal also bears subjective information (identity, quality, emotional state), which cannot be evaluated using the relations established by now [26].

Image modelling [17] [35]

As previously shown in example 2.2 the brightness of each black-and-white TV image element (pixel) is quantified with m levels of grey assumed to be equally probable and then the average quantity of information contained in a pixel, in the independent elements approximation, was:

$$D = H_{\max} = \log_2 m \text{ [bits/element pixel]}$$

In practice the pixels are not independent and the m levels of grey are not equally probable. If only the statistical relations between neighbouring elements are taken into consideration, the conditioned entropy will be:

$$H_1 = - \sum_{i=1}^m \sum_{j=1}^m p(a_i) p(a_j/a_i) \log_2 p(a_j/a_i)$$

where $p(a_i)$ represents the i -level probability, $p(a_j/a_i)$ is the probability of level j after level i .

Calculations of this entropy have shown that, for typical television images, and considering only the image space statistics, two times lower values on average are obtained compared to the case when modelling with a discrete memoryless source (the image elements are considered independent) and for which we have:

$$H_0 = - \sum_{i=0}^m p(a_i) \log_2 p(a_i) \text{ [bits/pixel]}$$

H_2 and H_3 have also been calculated; they correspond to 2nd and 3rd order Markov models. But it was found that the decrease in entropy for these models is too little, being hence unpractical.

A method for establishing conditional entropy is to process the correlated memory source in order to become independent (de-correlation). As we will see in chapter 3, statistical independence is a condition for an optimal coding.

A simple and efficient method for physical achievement of this condition is to use differential modulation [35] (see also 2.8).

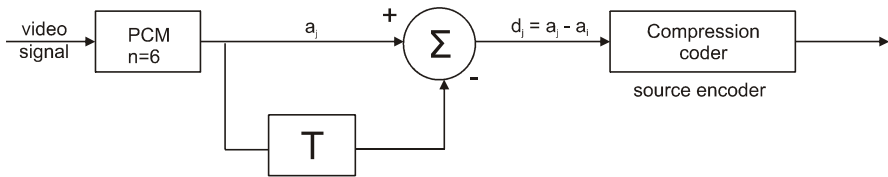


Fig. 2.16 Differential Pulse Code Modulation for a black-and-white video signal

The analogue black-and-white video signal is converted into a digital signal inside PCM (Pulse Code Modulation) block so that each sample will be transmitted using $n=5$ bits. Assuming that samples are independent and using entropy formula for a discrete memoryless source we obtain:

$$H(A) \cong 5 \text{ bits/sample .}$$

In fact video signal samples are strongly correlated, PCM output being in fact a memory source. For monochrome video signals it can be assumed that the differences between two successive samples d_j are approximately independent, therefore the entropy corresponding to this differences, calculated with relation (2.12) will be close to the conditioned entropy of PCM signal. In this case $H_d(A) \cong 3 \text{ bits/sample}$, a value twice lower than that of PCM source.

Remarks

- The actual value of source entropy is very important when trying to achieve source compression, as we will see in chapter 3.
- The differential transmission for colour video signals is thought similarly [35].
- Other important applications for Markov chains are:
 - synchronisation in digital telephony equipment [2], [29]
 - voice recognition [13], [28].

Memory channels

A transmission channel is a memory channel if the output signals depend on the previously transmitted ones.

Typical examples of memory channels are:

- radio channels with burst errors caused by fading phenomenon
- transmission wires and cables; transmitted sequences of symbols through the channel are affected by burst noise caused by switching and inter-modulation
- magnetic recording media where recording gaps may appear because impurities and dust particles.

Typical for these memory channels is the fact that noise does not affect independently each transmitted symbol, but burst errors occur and that is why these channels are also called *burst-error channels*.

We call a *burst error* a sequence of symbols (accurate or not) in which the first and last one are erroneous and the successive accurate symbols occur in groups smaller than r .

Parameters characterizing burst error are:

- r : between last erroneous symbol of a burst and the first erroneous symbol of the next burst there are more than r accurate successive symbols
- l : *burst error length* is given by total number of symbols (erroneous and accurate) forming the burst error
- D : *error density* is defined as the ratio between the number of erroneous symbols of the burst (t) and the length of the burst(l)

Here is a sample of a burst of length $l = 7$:

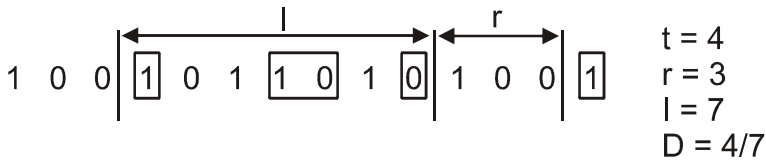


Fig. 2.17 Example of burst

Poisson law approximates burst error distribution. In order to obtain a better approximation of experimental curves, in channels where burst errors have a grouping tendency, other distribution laws have been searched for: hyperbolic law, Pareto distribution [1], [24].

A simplified model of a memory channel may be the one represented in Fig. 2.18 [14].

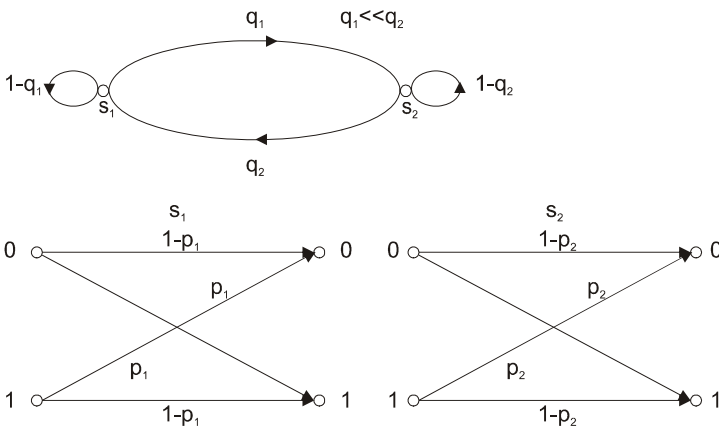


Fig. 2.18 The simplified model of a memory channel

This model has two states: s_1 and s_2 , s_1 being a “good state” in which there are few errors ($p_1=0$), and s_2 a “bad state” when errors occur very frequently ($p_2=1/2$). The channel remains in state s_1 almost all the time, its transition to state s_2 are caused by transmission characteristics modifications, for instance strong fading. As a consequence errors occur in bursts, because of the high value of p_2 ($p_2 \approx 0.5$).

The study of these memory channels presents great interest because of their applications in mobile communications and satellite transmissions.

References

- [1] Angheloiu, I.: Teoria Codurilor. Editura Militara, Bucuresti (1972)
- [2] Bellamy, J.C.: Digital Telephony, 2nd edn. Wiley, Chichester (1991)
- [3] Berrou, C., Glavieux, A.: Near Shannon limit error-correcting coding and decoding turbo-codes. In: Proc. ICC 1993, Geneva, pp. 1064–1070 (1993)
- [4] Blahut, R.E.: Digital Transmission of Information. Addison-Wesley, Reading (1990)
- [5] Cullmaun, G.: Codage et transmission de l’information. Editions Eyrolles, Paris (1972)
- [6] Fontolliet, P.G.: Systemes de telecommunícations. Editions Georgi, Lausanne (1983)
- [7] Gallager, R.G.: Information Theory and Reliable Communication. John Wiley & Sons, Chichester (1968)
- [8] Hamming, R.: Coding and Information Theory. Prentice-Hall, Englewood Cliffs (1980)
- [9] Haykin, S.: Communication Systems, 4th edn. John Wiley & Sons, Chichester (2001)
- [10] Ionescu, D.: Codificare si coduri. Editura Tehnica, Bucuresti (1981)
- [11] Iosifescu, M.: Lanturi Markov finite si aplicatii. Editura Tehnica Bucuresti (1977)
- [12] Kolmogorov, A.N.: Selected Works of A.N. Kolmogorov. In: Shirayev, A.N. (ed.) Probability Theory and Mathematical Statistics, 1st edn., vol. 2. Springer, Heidelberg (1992)
- [13] Kriouile, A.: La reconnaissance automatique de la parole et les modeles markoviens caches. These de doctorat. Universite de Nancy I (1990)
- [14] Lin, S., Costello, D.: Error control coding. Prentice-Hall, Englewood Cliffs (1983)
- [15] Mateescu, A., Banica, I., et al.: Manualul inginerului electronist, Transmisii de date. Editura Tehnica, Bucuresti (1983)
- [16] McEliece, R.J.: The Theory of Information and Coding, 2nd edn. Cambridge University Press, Cambridge (2002)
- [17] Mitrofan, G.: Televiziune digitala. Editura Academiei RSR, Bucuresti (1986)
- [18] Mihoc, G., Micu, N.: Teoria probabilitatilor si statistica matematica. Editura Didactica si Pedagogica, Bucuresti (1980)
- [19] Miller, M., Vucetic, B., Berry, L.: Satellite communications. In: Mobile and fixed Services. Kluwer Press, Dordrecht (1993)
- [20] Moon, T.K., Stirling, W.C.: Mathematical Methods and Algorithms for Signal Processing. Prentice-Hall, Englewood Cliffs (2000)
- [21] Murgan, A., et al.: Teoria Transmisiunii Informatiei. Probleme. Editura Didactica si Pedagogica, Bucuresti (1983)
- [22] Onicescu, O.: Probabilitati si procese aleatoare. Editura Stiintifica si Enciclopedica, Bucuresti (1977)

- [23] Oppenheim, A.V., Schafer, R.F.: Discrete-Time Signal Processing, 2nd edn. Prentice-Hall, Englewood Cliffs (1999)
- [24] Papoulis, A.: Probability, Random Variables and Stochastic Processes, 2nd edn. McCraw Hill (1994)
- [25] Peterson, W.W., Weldon, E.J.: Error-Correcting Codes, 2nd edn. MIT Press, Cambridge (1972)
- [26] Petrica, I., Stefanescu, V.: Aspecte noi ale teoriei informatiei. Editura Academiei RSR, Bucuresti (1982)
- [27] Proakis, J.: Digital Communications, 4th edn. Mc Gran-Hill (2001)
- [28] Rabiner, L., Schafer, R.W.: Introduction to Digital Speech Processing (Foundations and Trends in Signal Processing). Now Publishers Inc. (2007)
- [29] Radu, M.: Telefonie numerica. Editura Militara, Bucuresti (1988)
- [30] Reza, F.M.: An introduction to information theory (Magyar translation). Budapest (1966)
- [31] Shannon, C.E.: A Mathematical Theory Of Communication. Bell System Technical Journal 27, 379–423 (1948); Reprinted in Shannon Collected Papers, IEEE Press(1993)
- [32] Spataru, A.: Fondements de la theorie de la tr nsmission de l'information. Presses Polytechniques Romandes, Lausanne (1987)
- [33] Stark, H., Woods, J.W.: Probability and Random Processes with Applications to Signal Processing, 3rd edn. Prentice-Hall, Englewood Cliffs (2002)
- [34] Vucetic, B., Yuan, J.: Turbo codes. Kluver Academic Publishers Group, Dordrecht (2001) (2nd Printing)
- [35] Wade, G.: Signal Coding and Processing. Cambridge University Press, Cambridge (1994)
- [36] Wozencraft, J.W., Jacobs, I.M.: Principles of Communication Engineering. Waveland Press, Prospect Heights (1990)

Chapter 3

Source Coding

Motto: *To defeat yourself is the first
and the most beautiful of
all victories.*

Democritus

3.1 What Is Coding and Why Is It Necessary?

Information is rarely transmitted directly to the receiver, without any processing, due to the followings:

- source alphabet is different from channel one, therefore some adaptation of the source to the channel is needed (*information representation codes*).
- channel must be very efficiently used (close as possible to its full capacity), meaning that the source must be converted into an optimal one (p_{io}) (*statistical adaptation of the source to the channel* from information theory point of view); keeping in mind the coding theory, we may say that for an efficient use of the channel - in order to minimize transmission time and/or storage space - source compression is needed (*compression codes*).
- it is also necessary to adapt the source to the channel in order to match the spectrum to channels characteristics and also to ensure synchronization between transmitter and receiver (*base band codes / line codes*).
- information transmitted over a noisy channel is distorted by channel noise; this is why error detecting and correcting codes are used in error control procedures (*error control codes*).
- information confidentiality from unauthorized persons must be provided in some applications (*encryption*).

The need for source processing prior to transmission (or storage) is leading to the processing known as coding.

For a better understanding of coding, we will divide it into three main parts:

1. source coding (C_s) or coding for noiseless channels.
2. encryption (E), i.e. coding for keeping information confidentiality.
3. channel coding (C_c) – protection to channel noise (error control) and channel adaptation (base band codes).

In real transmission, one, two or all these aspects arise, the necessary processing degree being imposed by the application itself.

Coding is the process of finding a bijective correspondence between source (S) messages and *codewords set* (C) obtained using the alphabet X.

Be a discrete memoryless source S and the corresponding codewords set C:

$$S: \left(\begin{matrix} s_i \\ p_i = p(s_i) \end{matrix} \right), i = \overline{1, M}, \sum_{i=1}^M p_i = 1 \tag{3.1}$$

$$C: \left(\begin{matrix} c_i \\ p_i = p(c_i) \end{matrix} \right), i = \overline{1, M}, p(c_i) = p(s_i) \tag{3.2}$$

where c_i is a *codeword* and represents a finite row of symbols x_j belonging to code alphabet X, assumed memoryless too:

$$X: \left(\begin{matrix} x_j \\ p_j = p(x_j) \end{matrix} \right), j = \overline{1, m}, \sum_{j=1}^m p_j = 1 \tag{3.3}$$

By encoding, the initial source S is transformed into a secondary one X, entering the channel (Fig. 3.1)

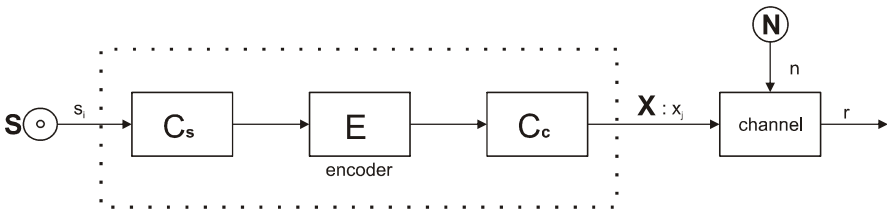


Fig. 3.1 Illustration of the transformation $S \rightarrow X$ realized through encoding

For a source S and a given alphabet X, we may create a multitude of codes. Table 3.1 shows some binary codes put into correspondence with a quaternary ($M = 4$) source S.

Table 3.1 Binary codes associated to a quaternary source ($M=4$)

S	A	B	C	D	E
S_1	00	0	0	0	0
S_2	01	01	10	10	100
S_3	10	011	110	110	11
S_4	11	0111	1110	111	110

Next, we will define some codes frequently used in applications:

- *Uniform code* is a code with same lengths of codewords (e.g.: code A).
- *Non-uniform code* is a code with different lengths of codewords (e.g.: codes B, C, D, and E).

- *Uniquely decodable code* (UDC) is a code with a unique succession of source symbols for each codewords succession (e.g.: A, B, C, D are UDC). Code E is a not a UDC because the codeword $c_4, 110$, can be decoded either s_4 or s_3s_1 .
- *Comma code* is a UDC using demarcation symbols between words (e.g.: B – ‘0’ indicates codewords beginning; C – ‘0’ indicates codewords ending).
- *Instantaneous code* (IC) is a UDC for which a codeword is not the prefix of another codeword (e.g.: A, C, D). Code B is not instantaneous because ‘0’ is prefix for other words.

Any code can be represented using the *coding tree* associated to codewords set. Such a tree is represented in Fig.3.2 for code D from table 3.1:

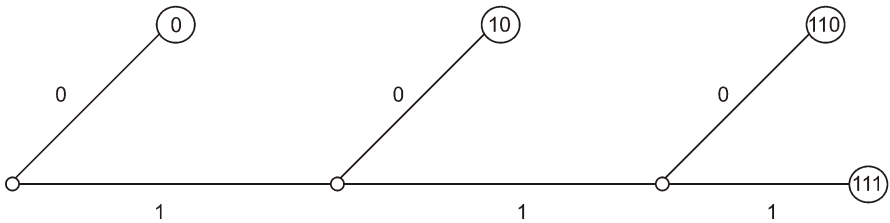


Fig. 3.2 Coding tree associated to code D from Table 3.1

3.2 Aim of Source Coding

As shown in section 3.1 source coding aim reduces to:

- adapting the source to the channel, if different as nature
- ensuring channel most rational use by *compression* (from coding theory point of view) and *statistical adaptation of the source to the channel* (from information theory point of view); the latter is achieved by maximizing the transinformation which implies equally probable symbols $x_j, p_{j0} = \frac{1}{m}, \forall j = \overline{1, m}$ for symmetric channels; therefore the channel must be used at its full capacity:

$$C_{\text{noiseless channel}} = H_{\text{max}}(X) = D(X) = \text{ld } m \tag{3.4}$$

3.3 Information Representation Codes

These codes are used for adapting the source to the channel as nature, in order to be able to transmit or store information.

3.3.1 Short History

Codes for information representation have been known since ancient times. Thus a series of symbols used in myths and legends, in heraldry and worship objects, are sending messages from long time-gone worlds.

We illustrate this by quoting the symbol of the olive tree:

”*Olive tree* [9]: A tree of many symbolic resources: peace, fertility, purity, power, victory and reward.... It has similar significance in all European and Oriental countries. In Rome, it was dedicated to Jupiter and Minerva. According to a Chinese legend, the olive tree wood would neutralize certain poisons and venom; therefore it was highly valued for its curing properties. In Japan it symbolizes amiability as well as success in studies and war: it is the tree of victory. In Jewish and Christian traditions the olive tree is a symbol of peace: the pigeon brought to Noah an olive tree branch at the end of the Great Flood. According to an ancient legend, the cross on which Jesus was crucified was manufactured from olive and cedar wood. Furthermore, in the Middle Ages language, it also stands for a symbol for gold and love. In Islam the olive tree is a central tree, the “axis mundi”, a symbol of the Universal Man, of the Prophet. Blessed tree, it is associated to Light, as the olive oil lightens the lamps... The olive tree symbolizes after all, the Heaven of the chosen ones.”

In ancient Greece [41], for remote messages transmission, a kind of “telegraph” was made using torches, without compression, as later in Morse code.

In 18th century, the Royal British Navy had used for transmission, signals on 6 bits, a system of cabins with 6 shutters and some lamps inside, which allowed the coding of $M = 2^6 = 64$ messages. These were used to encode 26 letters, 10 numbers and some other special commands, common words or phrases. In this way, they achieved, beside information representation, some sort of compression. In [41] it is shown that two of the 28 combinations, represented the command to execute or acquit a convict. It is mentioned the case of a convict executed due to a fatal transmission error of the message. This example emphasis compression weakness to error in transmission and/or storage.

The acronyms used since ancient times are, in fact, compressed ways of representing information. Thus, on Roman funeral graves, for which the engraving cost was very high, it is frequently met STL (Sit Tibi Terra Levis), acronym corresponding in English to “May the earth rest lightly on you”. Regarding acronyms utilization in present times, we can refer to our era as “civilization of acronyms”; each domain uses real dictionaries of acronyms. Lets take for example ITC (Information Theory and Coding), SR (Shift Register), AOC (Absolute Optimal Code) a. s. o. But same acronyms could represent many other things: Informational Trade Center, Security Report, Airline Operation Center etc., illustrating in fact its vulnerability to errors.

In what follows we will present some of the most used codes for information representation in data transmissions, analogue to digital converters including numbering systems and finally the genetic code, taking into account its universality and actuality.

3.3.2 Numeral Systems

A *numeral system* is a mathematical notation for representing numbers of a given set, using distinct symbols (a finite alphabet b).

There is a diversity of numeral systems developed from ancient time to modern days.

Unary system, is the simplest numeral system and it uses only one letter alphabet: $b=1$, for example a symbol x . The representation of n symbols is n times x (e.g. three is represented: xxx). It was used in early days of human civilization.

Sign – value system is a variation of unary system. It uses an enhanced alphabet by introducing different symbols for certain new values, for example for power of 10, $-$, for power of 100, $+$. The number 213 could be represented:

$$213 \rightarrow ++-XXX$$

The Egyptian numeral system was of this type and the Roman numeral system was a modification of this idea.

System using special abbreviations for repetitions of symbols; for example: A – one occurrence, B – two occurrence, ... , I – nine occurrence. In this case the number 205 will be written: B+EX.

Positional systems (place – value notation) using base b . A number N in base b is represented using b symbols (digits) corresponding to the first b natural numbers, including zero:

$$A = \{0, 1, 2, \dots, b-1\} = \{a_i\}, i = \overline{0, b-1} \quad (3.5)$$

$$N_b = (a_{n-1} \dots a_i \dots a_0)_b = a_{n-1}b^{n-1} + \dots + a_1b^1 + a_0b^0 \quad (3.6)$$

Fractions in the positional system are written dividing the digits into two groups:

$$(a_{n-1} \dots a_0, c_1 c_2 \dots)_b = \sum_{i=0}^{n-1} a_i b^i + \sum_{i=0}^{\infty} c_i b^{-i} \quad (3.7)$$

The numbers b^i and b^{-i} are the weights of the corresponding digits. The position i is the logarithm of the corresponding weight:

$$i = \log_b b^i \quad (3.8)$$

According to the value of b , a lot of different numeral systems can be obtained, the most used in today life and in data communications and computing being:

- *decimal numeral system* (dec) : $b=10$, with the alphabet:

$$A = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

- *octal numeral system* (oct) : $b=8$ and the alphabet:

$$A = \{0, 1, 2, 3, 4, 5, 6, 7\}$$

- *hexadecimal numeral system* (hex) : $b=16$ and the alphabet:

$$A = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$$

Conversion between bases: can be done using the method of successive division by b .

Table 3.2 Conversion between hex, dec, oct and binary numeral systems

N_b	hex	dec	oct	Binary			
				b_3	b_2	b_1	b_0
0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	1
2	2	2	2	0	0	1	0
3	3	3	3	0	0	1	1
4	4	4	4	0	1	0	0
5	5	5	5	0	1	0	1
6	6	6	6	0	1	1	0
7	7	7	7	0	1	1	1
8	8	8	10	1	0	0	0
9	9	9	11	1	0	0	1
10	A	10	12	1	0	1	0
11	B	11	13	1	0	1	1
12	C	12	14	1	1	0	0
13	D	13	15	1	1	0	1
14	E	14	16	1	1	1	0
15	F	15	17	1	1	1	1

Remark

The numeral systems used from ancient times to nowadays, show strong compression features, meaning less time in transmission and/or space in storage (writing, reading).

3.3.3 Binary Codes Used in Data Transmission, Storage or Computing

- **Morse Code**

Although this code invented by Samuel Morse in 1837 for electric telegraph is not anymore actual, it still remains the universal code of amateur or professional radio operators (maritime links), especially in manually operating systems.

Alphanumeric characters (26 letters of the Latin alphabet and 10 decimal numbers) are encoded using three symbols: dot, line and space. Morse alphabet also makes an ad-lib compression putting into correspondence the shortest words to letters with maximal frequency (from English language).

The tree corresponding to Morse alphabet is shown in Table3.3.

Table 3.3 Tree corresponding to Morse code

•	E	•	I	•	S	•	H
				—	U	—	V
			—	A	•	R	•
		—			W	—	Ä
		•		D	•	P	
		—	T	•	N	•	D
—	K					—	X
—	M				•	G	•
				—	O	—	Y
	•			G	•	Z	
—	O			—	Q		
•	O	•	Ö				
—	O	—	Ch				

The “SOS” message in Morse code is “. . . - - - . . .”

• Baudot Code

Morse code, a non-uniform code, has the drawback of a difficult automatic decoding. That is why Emile Baudot proposed in 1870 for telegraphic transmissions a five letters long uniform code (teleprinter code), known as ITA¹ / CCITT² / ITU³ code (Table. 3.4). The 56 characters (26 letters, 10 numbers, 12 signs and 8 commands) used in telegraphy cannot be uniquely encoded with 5 bits: $M = 2^5 = 32 < 56$. That is why the 56 character set was divided in two subsets: a lower set containing the letters and an upper set containing the numbers and other figures; commands are uniquely decodable.

The same binary sequence is assigned to two different characters, but belonging to different sets.

Any change from one set to the other, during a message, is preceded by an *escape codeword*:

1 1 1 1 1 indicates the lower set

1 1 0 1 1 indicates the upper set.

¹ ITA – International Telegraph Alphabet.

² CCITT – Comité Consultatif International Téléphonique et Télégraphique.

³ ITU – International Telecommunication Union.

Remark

The escape codes generally obtain a decrease in length of encoded message (therefore a compression), if there are not too many inter-sets changes in the message (condition achieved in most telegraphic transmissions).


Table 3.4 Baudot Code







The number of the combination	Letters	Numbers and special signs	Code combination	The number of the combination	Letters	Numbers and special signs	Code combination
1	A	-	11000	17	Q	1	11101
2	B	?	10011	18	R	4	01010
3	C	:	01110	19	S	;	10100
4	D	who are you	10010	20	T	5	00001
5	E	3	10000	21	U	7	11100
6	F	!	10110	22	V	=	01111
7	G	&	01011	23	W	2	11001
8	H	£	00101	24	X	/	10111
9	I	8	01100	25	Y	6	10101
10	J	ringer	11010	26	Z	+	10001
11	K	(11110	27	Carriage Return (CR)		00010
12	L)	01001	28	New Letter (NL)		01000
13	M	.	00111	29	Letter shift		11111
14	N	,	00110	30	Figure shift		11011
15	O	9	00011	31	Space (SP)		00010
16	P	0	01101	32	Unusable		00000

- **ASCII Code**

ASCII (American Standard Code for International Interchange) known also as CCITT 5 or ISO⁴ code is a 7 bit length code which allows letters, numbers and numerous special commands representation without escape code character as in Boudot code ($2^7 = 128 = 33$ non – printing (control) characters + 94 printable + 1 space). It was proposed in 1960 and from then it is the most used code in data encoding. The code is given in Table. 3.5

⁴ ISO - International Organization for Standardization.

Table 3.5 The 7-bit ISO code (CCITT N° 5, ASCII). Command characters:  - for national symbols, SP – Space, CR - Carriage Return, LF - Line Feed, EOT - End of Transmission, ESC – Escape, DEL - Delete.

				bit 7	0	0	0	0	1	1	1	1
				bit 6	0	0	1	1	0	0	1	1
				bit 5	0	1	0	1	0	1	0	1
bit 4	bit 3	bit 2	bit 1		0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0		P	'	p
0	0	0	1	1	SOH	DC ₁	!	1	A	Q	a	q
0	0	1	0	2	STX	DC ₂	”	2	B	R	b	r
0	0	1	1	3	ETX	DC ₃	#	3	C	S	c	s
0	1	0	0	4	EOT	DC ₄	␣	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	,	7	G	W	g	w
1	0	0	0	8	BS	CAN	(8	H	X	h	x
1	0	0	1	9	HT	EM)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K		k	
1	1	0	0	12	FF	FS	,	<	L		l	
1	1	0	1	13	CR	GS	-	=	M		m	
1	1	1	0	14	SO	RS	.	>	N	^	n	-
1	1	1	1	15	SI	US	/	?	O	-	o	DEL

Decimal format of ASCII code is easily obtained taking into consideration the weights of the bits; as example:

Letter s → binary: 1110011 → decimal: $2^6+2^5+2^4+2^1+2^0 = 115$

In many cases to the 7 data bits is added one more (the 8th bit), the parity control bit (optional); it results the ASCII-8 code able to detect odd errors (see 5.7.10).

• BCD Code

BCD (Binary Coded Decimal) code, known sometimes NBCD (Natural BCD) is a 4 – bits code for decimal numbers, in which each digit is represented by its own binary natural sequence. It allows easy conversion from decimal to binary for printing and display. This code is used in electronics in 7 – segments displays, as well in financial, commercial and industrial computing using decimal fixed point or floating – point calculus.

IBM (International Business Machines Corporation), in its early computers used a BCD – 6 bits length code (Table. 3.6)

Table 3.6 IBM BCD code – 6 bits length

Information (characters)	Code combination	Information (characters)	Code combination
0	00 0000	BLANK	01 0000
1	00 0001	/	01 0001
2	00 0010	S	01 0010
3	00 0011	T	01 0011
4	00 0100	U	01 0100
5	00 0101	V	01 0101
6	00 0110	W	01 0110
7	00 0111	X	01 0111
8	00 1000	Y	01 1000
9	00 1001	Z	01 1001
SPACE	00 1010	=	01 1010
=	00 1011	,	01 1011
,	00 1100	(01 1100
	00 1101	-	01 1101
√	00 1110	,	01 1110
>	00 1111	CANCEL	01 1111
-	10 0000	+	11 0000
J	10 0001	A	11 0001
K	10 0010	B	11 0010
L	10 0011	C	11 0011
M	10 0100	D	11 0100
N	10 0101	E	11 0101
O	10 0110	F	11 0110
P	10 0111	G	11 0111
Q	10 1000	H	11 1000
R	10 1001	I	11 1001
!	10 1010	?	11 1010
S	10 1011	.	11 1011
*	10 1100)	11 1100
	10 1101	[11 1101
;	10 1110	<	11 1110
Δ	10 1111	≠	11 1111

- **EBCDIC code**

EBCDIC (Extended Binary Coded Decimal Interchange Code) is an extension of the BCD code to 8 bits length being proposed by IBM in the operating systems of its computers in the years of 1963 – 1964 (Table 3.7).

Table 3.7 EBCDIC code

				b ₀	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
				b ₁	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	
				b ₂	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	
				b ₃	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	
b ₄	b ₅	b ₆	b ₇		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	0	0	0	0	NUL	DLE	DS		blk	&	-									0	
0	0	0	1	1	SOH	DC ₁	SOS			/			a	j			A	J		1	
0	0	1	0	2	STX	DC ₂	FS	SYN					b	k	s		B	K	S	2	
0	0	1	1	3	ETX	DC ₃							c	l	t		C	L	T	3	
0	1	0	0	4	PF	RES	BYP	PN					d	m	u		D	M	U	4	
0	1	0	1	5	HT	NL	LF	RS					e	n	v		E	N	V	5	
0	1	1	0	6	LC	BS	EOB	UC					f	o	w		F	O	W	6	
0	1	1	1	7	DEL	IDL	PRE	EOT					g	p	x		G	P	X	7	
1	0	0	0	8		CAN							h	q	y		H	Q	Y	8	
1	0	0	1	9		EM							i	r	z		I	R	Z	9	
1	0	1	0	10	SMM	CC	SM		Ç	!											
1	0	1	1	11	VT				.	\$,	>									
1	1	0	0	12	FF	IFS		DC ₄	<	*	%	@									
1	1	0	1	13	CR	IGS	ENQ	NAK	()	-	,									
1	1	1	0	14	SO	IRS	ACK		+	;	>	=									
1	1	1	1	15	SI	IUS	BEL	SUB		¬	?	“									

• **Gray Code**

In 1947, Frank Gray from Bell Laboratories introduced the term *reflected binary code* in a patent application, based on the fact that it may be built up from the conventional binary code by a sort of reflexion process.

The main feature of this code is that a transition from one state to a consecutive one, involves only one bit change (Table 3.8).

The conversion procedure from binary natural to Gray is the following: the most significant bit (MSB) from the binary code is the same with the MSB from the Gray code. Starting from the MSB towards the least significant bit (LSB) any bit change (0→1 or 1→0) in binary natural, generates an '1' and any lack of change generates a '0', in Gray code.

The conversion from Gray to binary natural is the reverse: the MSB is the same in binary natural code as well as in Gray code. Further on, from MSB to LSB, the next bit in binary natural code will be the complement of the previous bit if the corresponding bit from Gray code is 1 or it will be identical with the previous bit if the corresponding bit from Gray code is 0.

Table 3.8 Binary natural and Gray 4 bits length codes representation

Decimal	Binary Natural Code (BN)				Gray Code	Decimal	Binary Natural Code				Gray Code						
	B ₃	B ₂	B ₁	B ₀			G ₃	G ₂	G ₁	G ₀		B ₃	B ₂	B ₁	B ₀	G ₃	G ₂
0	0	0	0	0	0	0	0	0	8	1	0	0	0	1	1	0	0
1	0	0	0	1	0	0	0	1	9	1	0	0	1	1	1	0	1
2	0	0	1	0	0	0	1	1	10	1	0	1	0	1	1	1	1
3	0	0	1	1	0	0	1	0	11	1	0	1	1	1	1	1	0
4	0	1	0	0	0	1	1	0	12	1	1	0	0	1	0	1	0
5	0	1	0	1	0	1	1	1	13	1	1	0	1	1	0	1	1
6	0	1	1	0	0	1	0	1	14	1	1	1	0	1	0	0	1
7	0	1	1	1	0	1	0	0	15	1	1	1	1	1	0	0	0

Gray code is used in position encoders (linear encoders, rotary encoders) that generate numerical code corresponding to a certain angle. Such transducer is made up with optical slot disks and for this reason it is impossible to modify simultaneously all the bits that could change between two consecutive values. This explains why the optical system that generates the number corresponding to a certain angle is encoded in Gray.

In fast converters that generate continuous conversions, Gray code is used due to the same reasons. For these converters the problem that arises is storing the results; if the storage command arrives before all bits settle at final value, the errors in binary natural code are great, whereas the Gray code maximal error equals the LSB value.

This code is also frequently used in modulation systems. Let us suppose we deal with an 8 levels amplitude modulation. Thus amplitude levels are assigned to each three bits, levels which are then transmitted in sequences. Let us see two examples, one with the binary natural code and other one using the Gray code (Table 3.9):

Table 3.9 Example of 3 bit code in BN and Gray representation

BN Code	Assigned Level	Gray Code
0 0 0	1V	0 0 0
0 0 1	2V	0 0 1
0 1 0	3V	0 1 1
0 1 1	4V	0 1 0
1 0 0	5V	1 1 0
1 0 1	6V	1 1 1
1 1 0	7V	1 0 1
1 1 1	8V	1 0 0

When errors occur, the transition to an adjacent level is most likely to occur. For example: a 4V level is transmitted and the receiver shows 5V. Using binary natural code we have 3 errors whereas in case of Gray code we have only one error that can easily be corrected with one error-correcting code.

3.3.4 Pulse Code Modulation (PCM)

The concept of PCM was given in 1938 by Alec Reeves, 10 years before Shannon theory of communications and transistor invention, too early to demonstrate its importance. This is the *basic principle of digital communications* which involves an analog to digital conversion (ADC) of the carrying information signal $x(t)$. The generated signal is a digital one, characterized by a decision rate (bit rate), thus PCM is an information representation code; it is a digital representation of an analog signal $x(t)$.

PCM generation is illustrated by the block scheme given in Fig 3.3.

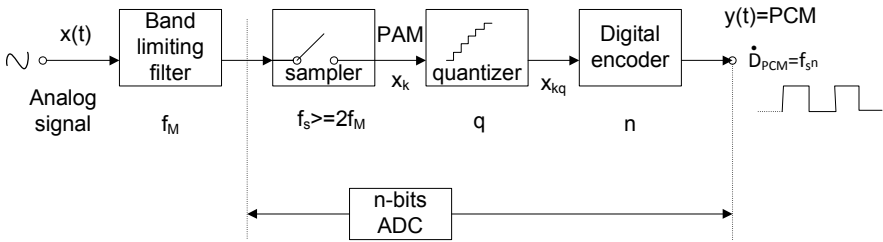


Fig. 3.3 Block scheme illustrating the generation of PCM signal (PCM modulator and coder)

At the receiver the processing is reversed, as illustrated in Fig 3.4 and represents a digital to analog conversion (DAC) followed by a low pass filtering (LPF) of recovered samples x_{kq} .

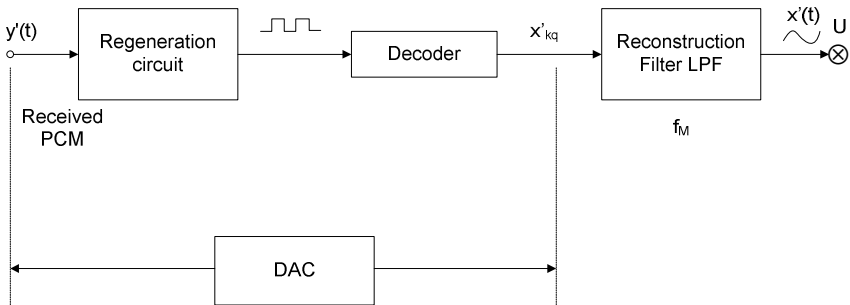


Fig. 3.4 Block scheme illustrating the receiving PCM process (PCM demodulator / decoder)

Remark

In transmission, the distorted PCM signal is regenerated using regenerative repeater, allowing the complete regeneration of the digital signal for very small SNR (<15dB), without cumulative distortions, as in analog transmissions (main advantage of digital).

As illustrated in Fig 3.3, the generation of PCM implies:

- low-pass filtering the analog signal $x(t)$ by an antialiasing analog filter whose function is to remove all the frequencies above f_M (the maximum frequency of $x(t)$)
- sampling the LP filtered signal at a rate a $f_S \geq 2 f_M$ (sampling theorem); in this point, the analog signal $x(t)$ is discretized in time, resulting PAM (Pulse Amplitude Modulation) signal x_k being the values of $x(t)$ at discrete moments kT_S , where, $T_S = 1/f_S$ is the sampling period.
- quantization: each sample x_k is converted into one of a finite number of possible values of the amplitude (q signifies the number of quantization levels) x_{kq}
- encoding: the quantized samples x_{kq} are encoded using a binary alphabet ($m=2$), each quantized sample being represented, usually, by a binary code word of length n , called PCM word.

Thus, the infinite number of possible amplitude levels of the sampled signal is converted into a finite number of possible PCM words. If n represents the length of PCM word, the total number of possible distinct words that can be generated (possible amplitude values of the quantized samples) are:

$$q=2^n \tag{3.9}$$

An example of PCM generation is illustrated in Fig 3.5.

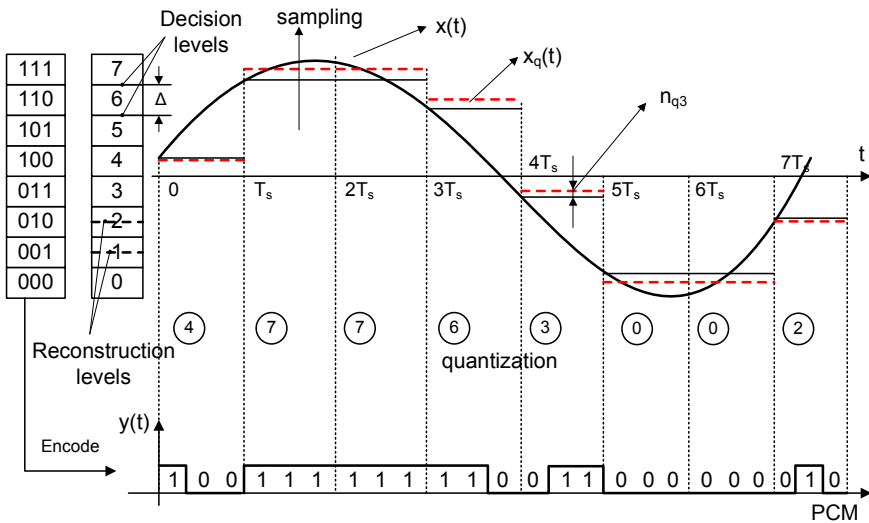


Fig. 3.5 Example of PCM generation

We may ask: which are the causes of distortions in PCM systems? If the sampling frequency is $f_S \geq 2f_M$, theoretically, sampling do not introduce distortions (in reality, this is not true, because the sampling is not ideal, Dirac distributions being impossible to be realized). Quantization, always introduces distortions because the real amplitude of a sample is quantized by a unique amplitude value (the reconstruction level), generally being the average value of the quantized interval. The difference between the real value of a sample and its quantized value is called *quantization noise*: n_q .

$$n_q =: x(kT_S) - x_q(kT_S) \tag{3.10}$$

The *step size* (quantum) Δ , under the assumption of uniform quantization ($\Delta=ct$), is:

$$\Delta_u = \frac{X}{q} \tag{3.11}$$

for unipolar signal : $x(t) \in [0, X]$, and

$$\Delta_b = \frac{2X}{q} \tag{3.12}$$

for polar signal : $x(t) \in [-X, X]$, where $|X|$ is the maximum level of $x(t)$ and q indicates the quantization levels.

If the number of quantization levels is high (fine quantization) the probability density function (pdf) of the quantization noise, $f(n_q)$, is considered uniform. (Fig 3.6)

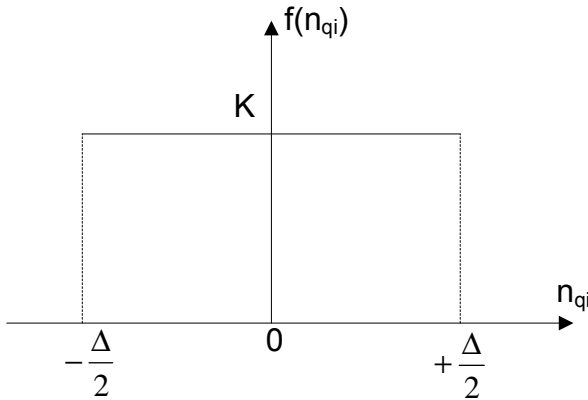


Fig. 3.6 Representation of quantisation noise pdf

In Fig 3.6 the index i is designating the i -th quantization interval.

Since, any random sample lies somewhere between $-\Delta/2$ and $+\Delta/2$ with probability 1, the pdf must satisfy:

$$\int_{-\Delta/2}^{\Delta/2} f(n_{qi}) dn_{qi} = 1 \tag{3.13}$$

For a constant pdf:

$$f(n_q)=K \quad (3.14)$$

we obtain:

$$f(n_{qi})=K=1/\Delta \quad (3.15)$$

Encoding means to express in a binary code the quantized samples . The code could be: Binary Natural (BN), Binary Coded Decimal (BCD), Gray etc.

If the quantization levels q are chosen, the length of the PCM word is:

$$n=\text{ld } q \in \mathbb{Z} \quad (3.16)$$

\mathbb{Z} meaning the integer set.

Concluding, PCM generation implies:

- band limiting filtering
- sampling of $x(t)$ with $f_s \geq 2f_M$
- quantization of samples with q levels
- encoding (for uniform encoding) the q levels using words of length $n=\log_2 q \in \mathbb{Z}$

Thus, an analog signal $x(t)$, band limited f_M , is transformed into a binary stream of bit rate:

$$\dot{D}_{\text{PCM}} = f_s \cdot n \quad (3.17)$$

PCM is thus an information representation code: the analog signal $x(t)$ is represented through PCM as a binary stream $\dot{D}_{\text{PCM}} = f_s \cdot n$:

$$x(t), f_M \xrightarrow{\text{ADC}} \dot{D}_{\text{PCM}} = f_s \cdot n$$

The *main advantage* of PCM is its great immunity to noise: the required SNR in transmission is much lower compared to analogue transmission because at the receiver, the decision concerns only two levels and not infinity as in the analogue case. Beside this, digital offers many other facilities: compression, error control, encryption etc.

For example: a SNR of 50dB at the user, in telephony is ensured with only approximately 15dB in transmission using PCM (for aprox. 15dB of SNR, a BER of 10^{-5} is obtained without error control coding – see Appendix C).

The *main disadvantage* of PCM is the increase of the requested bandwidth. Example: for telephony, the bandwidth is [300, 3400]Hz; in digital telephony, the sampling frequency is $f_s = 8\text{KHz}$ and the length of the PCM word is $n=8$. Consequently, the PCM bit rate is: 64Kbps. The required bandwidth (baseband) is, according to Nyquist theorem: $B_{\text{PCM}} \approx 0,8 \cdot \dot{D}_{\text{PCM}} = 51,12\text{KHz}$, which is almost twenty times greater than that one corresponding to the analog signal (3,1KHz).

Despite this disadvantage, which can be reduced, by processing (compression, modulation), PCM is widely used being the basic principle in digital communications (transmission and storage).

The main advantages of the digital, compared to the analog, are:

- high quality transmission/storage in channel/storage media with very poor signal/noise ratio (SNR)
- multiple access by multiplexing (time division multiplex access - TDMA or code division multiplex access - CDMA)
- digital facilities as: compression, error control coding, encryption etc.

Noise in PCM systems

In PCM systems, the noise has two main sources: quantization (during PCM generation) and decision (made at the receiver, when a decision is made concerning the received value: 0 or 1)

a) *Quantization noise* [12]

This noise is defined by relation (3.10):

$$n_q =: x(kT_S) - x_q(kT_S) \quad (3.10)$$

The power of the quantizing noise N_q is:

$$N_q = \frac{1}{R} \overline{n_q^2} \quad (3.18)$$

$\overline{n_q^2}$ meaning the expected value of the noise power for q quantizing levels:

$$\overline{n_q^2} = \sum_{i=1}^q \int_{-\Delta/2}^{\Delta/2} n_{qi}^2 \cdot f(n_{qi}) dn_{qi} P_i \quad (3.19)$$

where P_i is the probability that the sample x_k falls within the i -th interval. Assuming that all the intervals are equally probable, we have:

$$P_i = \frac{1}{q}, \quad \forall i = \overline{1, q} \quad (3.20)$$

Replacing $f(n_{qi})$ with (3.15) in (3.19), we obtain:

$$\overline{n_q^2} = \sum_{i=1}^q \left(\int_{-\Delta/2}^{\Delta/2} n_{qi}^2 dn_{qi} \right) \cdot \frac{1}{\Delta} \cdot \frac{1}{q} = \frac{\Delta^3}{12} \cdot \frac{1}{\Delta} \cdot q \cdot \frac{1}{q} = \frac{\Delta^2}{12} \quad (3.19.a)$$

Consequently the power of the quantizing noise for uniform quantization is:

$$N_q = \frac{1}{R} \cdot \sum_{i=1}^q \int_{-\Delta/2}^{\Delta/2} n_{qi}^2 \cdot f(n_{qi}) dn_{qi} P_i = \frac{1}{R} \cdot \frac{\Delta^2}{12} \quad (3.18.a)$$

The quantizing SNR is:

$$\text{SNR}_q = \frac{P_S}{N_q} \quad (3.21)$$

P_S defining the signal power:

$$P_S = \frac{1}{R} X_{\text{rms}}^2 \quad (3.22)$$

which finally gives:

$$\text{SNR}_q = 12 \frac{X_{\text{rms}}^2}{\Delta^2} \quad (3.21.a)$$

Using (3.21.a), some other formulae can be obtained for SNR_q . Replacing Δ with relation (3.12) – the bipolar case, we have:

$$\text{SNR}_q = 12 \frac{X_{\text{rms}}^2}{4X^2} = 3 \left(\frac{X_{\text{rms}}}{X} \right)^2 \cdot q^2 \quad (3.21.b)$$

If q is replaced with (3.9), (3.21.b) becomes:

$$\text{SNR}_q = 3 \left(\frac{X_{\text{rms}}}{X} \right)^2 \cdot 2^{2n} \quad (3.21.c)$$

Defining as *crest factor* C the ratio:

$$C = \frac{X}{X_{\text{rms}}} \quad (3.23)$$

it is possible to write:

$$\text{SNR}_q = 3 \frac{q^2}{C^2} \quad (3.21.d)$$

If the SNR_q is expressed in dB, the relation (3.21.c) can be written as:

$$\text{SNR}_q [\text{dB}] = 4,7 \text{dB} + 6n + 20 \lg \frac{X_{\text{rms}}}{X} \quad (3.21.e)$$

Relation (3.21.e) is very practical, indicating that each supplementary bit of a PCM word enhance with 6dB the SNR_q .

Remarks

Relation (3.21.a) shows that for an uniform quantization ($\Delta=ct$), the quantization SNR is signal dependent, being an important disadvantage of uniform quantizers. The solutions to make a SNR_q signal $x(t)$ independent, are:

- non-uniform quantization, meaning the use of small steps ($\Delta \downarrow$) for weak signals ($x(t) \downarrow$) and large steps ($\Delta \uparrow$) for strong signals ($x(t) \uparrow$)
- companding the signal dynamic range: it makes a compression before quantization and an expansion after DAC in order to obtain the original dynamic range (Fig. 3.7)

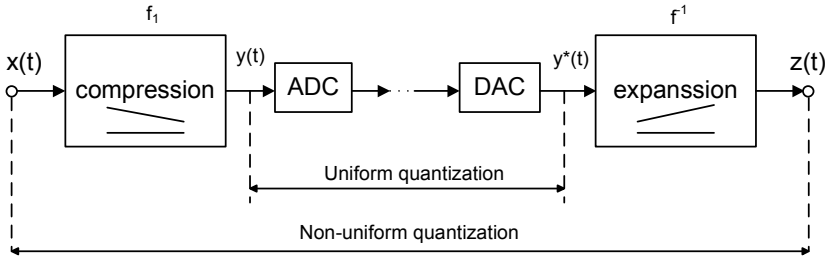


Fig. 3.7 Companding illustration

Companding means:

- compression

$$y = f_1(x) \tag{3.24}$$

- uniform quantization with q intervals

$$y^* = f_2(y) = f_2(f_1(x)) \tag{3.25}$$

- extension:

$$z = f^{-1}(y^*) = f^{-1}(f_2(f_1(x))) = f_3(x), \tag{3.26}$$

representing the non-uniform quantization characteristic. The quantization step Δ is signal dependent and is given by the compression characteristic:

$$\frac{dx}{dy} = \frac{\Delta_o}{\Delta(x)} \tag{3.27}$$

An ideal non-uniform quantization implies:

$$\Delta(x) = ct|x| \tag{3.28}$$

Replacing $\Delta(x)$ in (3.27), we have:

$$dy = \frac{\Delta_o}{ct|x|} dx = K \cdot \Delta_o \cdot \frac{dx}{|x|} \tag{3.29}$$

This differential equation has as solution a logarithmic characteristic

$$|y| = X + K\Delta_o \cdot \lg \frac{|x|}{X}, \tag{3.30}$$

graphically represented in Fig 3.8.

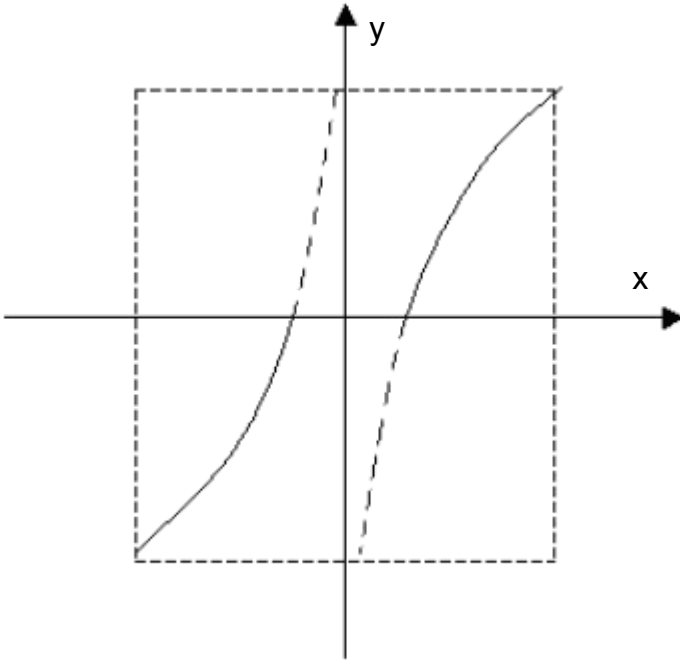


Fig. 3.8 Ideal compression characteristic

Remark

The characteristic (3.30) is impossible to be practically implemented because if $x \rightarrow 0 \Rightarrow y \rightarrow \infty$. The technical solutions for a non uniform quantization are approximations of this ideal characteristic.

In PCM systems, the compression characteristics (standards) are [3]: A law – for Europe, μ law – for North America, Japan, with the corresponding characteristics:

$$y = \begin{cases} \frac{\Delta x}{1 + \ln A}, & 0 \leq x \leq \frac{1}{A} \\ \frac{1 + \ln Ax}{1 + \ln A}, & \frac{1}{A} \leq x \leq 1 \end{cases} \quad (3.31)$$

$$y = \frac{\ln(1 + \mu x)}{\ln(1 + \mu)} \quad (3.32)$$

b) *Decision noise*

This noise occurs if PCM words are erroneous, meaning that the corresponding quantized samples will be erroneous too. We will discuss this problem under the following hypotheses:

- independent errors
- p is the BER of the channel.

- the noise corrupts only one bit in a word of length n (fulfilled in usual application)
- the code is BN (the worst from this point of view, but the most used)

Example 3.1

Be the correct PCM word: (1 1 1 1), the first left bit being MSB.

The corresponding quantized sample is:

$$(1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0) \cdot \Delta = 15V$$

if Δ is assumed to be 1V.

Assume that 1 error occurs in a four bits length; the errors are very different, according the affected bit, as follows:

Received word		x_{kq}	Error
MSB	LSB		
1	1 1 0	14V	1V
1	1 0 1	13V	2V
1	0 1 1	11V	3V
0	1 1 1	7V	8V

Decision noise n_d is defined as:

$$n_d = x_{kq} - x_{kq}' \quad (3.33)$$

where x_{kq} designates the correct quantized sample and x_{kq}' the erroneous one.

Under the given hypotheses, we have the probability of the erroneous word:

$$p_w = C_n^i p(1-p)^{n-1} \approx np \quad (3.34)$$

Decision noise if the error is situated on the i -th position, $i = \overline{0, n-1}$, is

$$\begin{cases} x_{kq} = (a_{n-1}q^{n-1} + \dots + a_i 2^i + \dots + a_0 2^0) \cdot \Delta \\ x_{kq}' = (a_{n-1}q^{n-1} + \dots + \overline{a_i} 2^i + \dots + a_0 2^0) \cdot \Delta \end{cases} \quad (3.35)$$

$$n_{di} = (a_i - \overline{a_i}) 2^i \cdot \Delta \quad (3.36)$$

Decision noise power N_d is:

$$N_d = \frac{1}{R} \overline{n_d^2} p_w \quad (3.37)$$

where $\overline{n_d^2}$ means the expected value of the decision noise power:

$$\overline{n_d^2} = \sum_{i=0}^{n-1} n_{di}^2 f(n_{di}) \quad (3.38)$$

where $f(n_{di})$ indicates the pdf of the decision noise, assumed equally probable for each position:

$$f(n_{di}) = \frac{1}{n} \quad (3.39)$$

Based on (3.36), (3.38), and (3.31), we obtain:

$$\overline{n_d^2} = \frac{1}{n} \sum_{i=0}^{n-1} 2^{2i} \Delta^2 = \frac{\Delta^2}{n} \sum_{i=0}^{n-1} 2^{2i} = \frac{2^{2n} - 1}{3} \cdot \frac{\Delta^2}{n}$$

and

$$N_d = \frac{1}{R} \cdot \frac{\Delta^2}{n} \cdot \frac{2^{2n} - 1}{3} \cdot np \quad (3.37.a)$$

or

$$N_d = \frac{1}{R} \cdot p \cdot \frac{\Delta^2}{3} (q^2 - 1) \approx \frac{1}{R} \cdot p \cdot \frac{\Delta^2}{3} \cdot q^2 \quad (3.37.b)$$

The total noise in the PCM systems is:

$$N_t = N_q + N_d = \frac{1}{R} \left(\frac{\Delta^2}{12} + p \frac{\Delta^2}{3} q^2 \right); \quad (3.40)$$

with the mention that the two noise sources are independent.

The total SNR in the PCM systems is:

$$\text{SNR}_t = \frac{S}{N_t} = \frac{\frac{1}{R} X_{\text{rms}}^2}{\frac{1}{R} \left(\frac{\Delta^2}{12} + p \frac{\Delta^2}{3} q^2 \right)} = \frac{12}{\Delta^2} \frac{X_{\text{rms}}^2}{1 + 4pq^2}; \quad (3.41)$$

If we replace Δ with (3.12) we obtain:

$$\text{SNR}_t = 3 \frac{q^2}{C^2 (1 + 4pq^2)}; \quad (3.42)$$

Remarks

- In PCM systems the total SNR (the detected SNR) is determined by:
 - a) quantization noise and
 - b) decision noise
- The quantization noise in PCM is word length (n) dependent and can be reduced by increasing the number of bits. Decision noise is channel dependent and can be reduced by increasing the SNR input in the channel (SNR_i). At a low level of SNR_i , the value of p is large and SNR_t is dependent of the SNR_i . This is the *decision noise limited region*, operating under a threshold ($\text{SNR}_{i0} \approx 15 \text{ dB}$). Over this threshold p is negligible and SNR_t is limited to a value given by the quantization noise, based on the PCM codeword length. This is the *quantization noise limited region* (see Fig 3.9).

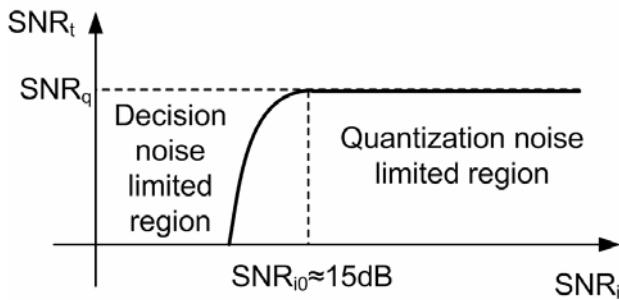


Fig. 3.9 Threshold effect in PCM systems

- Quantization noise in PCM cannot be eliminated, being linked to the quantization process. The reconstructed signal $x'(t)$, obtained after the final lowpass filtering of the received quantized samples (x_{kq}) assumed error free in transmission (without decision noise) is only an approximation of the original signal $x(t)$. The difference $x(t) - x'(t)$ is called *distortion*. It can be reduced if n , the length of PCM word, is increased, which implies bandwidth increase also. A trade is made between fidelity and quality, known as the *acceptable loss* and described by *rate distortion theory* [4].

Example 3.2

Consider a 1kHz sinusoidal voltage of 1V amplitude. Determine the minimum number of quantizing levels required for a SNR of 33dB. Which is the length of the corresponding PCM word, assuming that in transmission BER is 10^{-2} . Calculate the total SNR.

Solution

The signal:

$$x(t) = X \sin 2\pi f t = \sin 2\pi 10^3 t$$

The quantizing SNR given by (3.23.e) is:

$$\begin{aligned} \text{SNR}_q [\text{dB}] &= 4.7 \text{dB} + 6n + 20 \lg \frac{X_{\text{rms}}}{X} = \\ &= 4.7 + 6n + 20 \lg \frac{1/\sqrt{2}}{1} = 33 \Rightarrow \\ 1.6 + 6n &= 32 \Rightarrow n = \frac{31.4}{6} \cong 5 \end{aligned}$$

$$f_s = 2f_M = 2 \text{kHz};$$

$$\bullet \text{D}_{\text{PCM}} = f_s n = 2 \cdot 10^3 \cdot 5 = 10 \text{kbps}$$

Based on (3.40) we have:

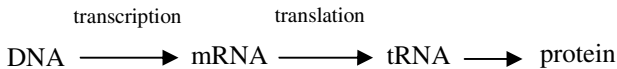
$$\text{SNR}_t = 3 \frac{q^2}{c(1+4pq^2)} = 3 \frac{2^{2n}}{2(1+4p2^{2n})} = 3 \frac{2^{10}}{2(1+4 \cdot 10^{-2} \cdot 2^{10})} = 15.64 \text{dB}$$

3.3.5 Genetic Code

Genetic information, meaning all the set of instructions necessary and sufficient to create life, is stored in the DNA (*DeoxyriboNucleic Acid*) of a cell [7].

Gene, located in DNA, is the basic unit of heredity and contains both coding sequences (exons) and non-coding sequences (introns), that determine when a gene is active (expressed).

Gene expression [32] is given by the *central dogma of molecular biology* (given by Francis Crick in 1958):



DNA, is a huge molecule (macro-molecule of more than $3,2 \cdot 10^9$ bp long for homo-sapiens), composed of two twisted complementary strands of nucleotides, linked in a double-helix structure, based on the hydrogen bonds: A=T and G≡T. This structure was discovered by J. Watson, F. Crick, M. Wilkins and Rosalind Franklyn in 1953 and priced with Nobel Prize in 1962.

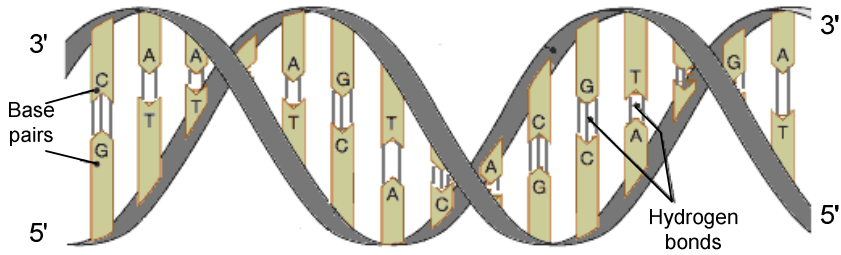


Fig. 3.10 DNA structure

DNA is a very stable medium, allowing long term storage of genetic information, based on the lack of hydroxyl group OH and on the complementarity of bonds $A=T$, $G\equiv T$, offering thus error control properties (if a mutation occurs in one strand, it could be corrected based on the complementarity of strands).

A *nucleotide* is composed of three groups:

- a phosphate P group, linked to
- a pentose sugar (D-deoxyribose) linked to
- one of four nitrogenous N base: A – Adenine, C – Cytosine, G – Guanine and T – Thymine.

The four nucleotides making DNA have different bodies, but all have the same pair of terminations: 5' - P – phosphoryl and 3' – OH – hydroxyl.

The nucleotide stream is encoding the information necessary to generate 20 distinct *amino-acids*, which are the building blocks of proteins. The identity of a protein is given not only by its composition, but also by the precise order of its constituent amino-acids.

RNA (RiboNucleicAcid) is a polymer very similar to DNA, but having some main differences:

- sugar deoxyribose (D) is replaced with ribose (R); it explains the great instability of RNA: alcohol groups OH are highly instable to enzymes, allowing a quick adaptation of RNA molecule to the environment stimuli (hormones, nutrients). This explains its use for gene expression.
- base Thymine (T) is replaced by Uracil (U);
- its structure is simple stranded, having the capacity of making secondary structures like DNA (tRNA - transfer RNA), if the simple strand has polydromic structure (that reads the same in both directions) regions [10].

During *transcription*, when a gene is expressed, the coding information found in exons is copied in messenger RNA (mRNA).

By *translation*, the genetic information of mRNA is decoded, using ribosomes as reading units, each codon being translated into an aminoacid (tRNA) and an aminoacid chain is giving a protein.

Thus genetic code is defined either as DNA or RNA, using base T or U respectively.

Proteins are macromolecules of 100 – 500 amino-acids known as residuus.

In order to be able to encode 20 distinct aminoacids, using a 4 letter alphabet (A, T/U, C, G), a minimum 3 nucleotides sequence, called *codon* is necessary ($4^2=16<20$, $4^3=64>20$).

Genetic code [10] defines the mapping between the 64 codons and amino-acids. Table 3.10 represents the encoding table: codons – amino-acids and Table 3.11 the decoding table: amino-acids – codons.

Table 3.10 Genetic code – encoding table

	U	C	A	G
	UUU Phe (F)	UCU Ser (S)	UAU Tyr (T)	UGU Cys (C)
U	UUC Phe (F)	UCC Ser (S)	UAC Tyr (T)	UGC Cys (C)
	UUA Leu (L)	UCA Ser (S)	UAA Stop	UGA Stop
	UUG Leu (L)	UCG Ser (S)	UAG Stop	UGG Trp (W)
	CUU Leu (L)	CCU Pro (P)	CAU His (H)	CGU Arg (R)
C	CUC Leu (L)	CCC Pro (P)	CAC His (H)	CGC Arg (R)
	CUA Leu (L)	CCA Pro (P)	CAA Gln (Q)	CGA Arg (R)
	CUG Leu (L)	CCG Pro (P)	CAG Gln (Q)	CGG Arg (R)
	AUU Ile (I)	ACU Thr (T)	AAU Asn (N)	AGU Ser (S)
A	AUC Ile (I)	ACC Thr (T)	AAC Asn (N)	AGC Ser (S)
	AUA Ile (I)	ACA Thr (T)	AAA Lys (K)	AGA Arg (R)
	AUG Met (M)	ACG Thr (T)	AAG Lys (K)	AGG Arg (R)
	GUU Val (V)	GCU Ala (A)	GAU Asp (D)	GGU Gly (G)
G	GUC Val (V)	GCC Ala (A)	GAC Asp (D)	GGC Gly (G)
	GUA Val (V)	GCA Ala (A)	GAA Glu (E)	GGA Gly (G)
	GUG Val (V)	GCG Ala (A)	GAG Glu (E)	GGG Gly (G)

Table 3.11 Genetic code – decoding table

Ala/A	GCU, GCC, GCA, GCG	Leu/L	UUA, UUG, CUU, CUC, CUA, CUG
Arg/R	CGU, CGC, CGA, CGG, AGA,AGG	Lys/K	AAA, AAG
Asn/N	AAU, AAC	Met/M	AUG
Asp/D	GAU, GAC	Phe/F	UUU, UUC
Cys/C	UGU, UGC	Pro/P	CCU, CCC, CCA, CCG
Gln/Q	CAA, CAG	Ser/S	UCU, UCC, UCA, UCG, AGU, AGC
Glu/E	GAA, GAG	Thr/T	ACU, ACC, ACA, ACG
Gly/G	GGU, GGC, GGA, GGG	Trp/W	UGG
His/H	CAU, CAC	Tyr/Y	UAU, UAC
Ile/T	AUU, AUC, AUA	Val/V	GUU, GUC, GUA, GUG
START	AUG	STOP	UAA, UGA, UAG

Legend: A – Alanine, F – Phenylalanine, S – Serine, Y – Tyrosine, C – Cysteine, L – Leucine, W – Tryptophan, P – Proline, H – Histidine, R – Arginine, Q - Glutamine, I – Isoleucine, T – Threonine, N – Asparagine, K – Lysine, M - Methionine, V – Valine, D – Aspartic acid, G – Glycine, E – Glutamic acid.

Table 3.9 and 3.10 represent the standard canonical genetic code. It is not universal: there are organisms where the synthesis of proteins relies on a genetic code that varies from the standard one. The discovery of genetic code, based on a triplet codon brought to A. Sanger the second Nobel Prize in 1970.

The main features of the genetic code are:

- *Redundancy* or *degeneracy*: the same aminoacid can be encoded by distinct codons: *e.g.* Alanine/A is specified by codons: GCU, GCC, GCA and GCG.
 - ✓ This situation defines the *four fold degenerate site*: the third position can be any of the four possible (A, C, G, U) and all the four codons are giving the same aminoacid.
 - ✓ A *two-fold degenerate site* is defined if only two of four possible nucleotides in that position give the same aminoacid.
 - E.g.*: Asparagine (Asn/N) is specified by: AAU, AAC (two pyrimidines C/U)
 - Aspartic acid (Asp/D) is specified by: GAU, GAC (two purines A/G)
 - ✓ *Three-fold degenerate site* is defined when changing 3 of four nucleotides, the same aminoacid is obtained.
 - E.g.*: (the only one) Isoleucine (Ile/I) – AUU, AUC, AUA.
 - ✓ *Six codons define the same aminoacid*:
 - e.g.*: Arg/R, Ser/S, Leu/L.
 - ✓ *Three STOP codons*: UAA, UGA, UAG called also *termination* or *non-sense* codons, which signal ending of polypeptide (protein) generated by translation.
 - ✓ Only two codons define a unique aminoacid: AUG – Methionine which also specify the START of translation and UGG – Tryptophane.
- From coding point of view, redundancy means that the number of code words (codons): $4^3 = 64$ is larger than the number of the messages to be encoded: $M = 20 \text{ amino-acids} + 1 \text{ START} + 1 \text{ STOP} = 22$.
- The difference: $64 - 22 = 42$ combinations (codons) are redundant and define the degeneracy of the code. Redundancy (see Cap. 5) allows error protection. *Degeneracy* of the genetic code makes it *fault-tolerant for point mutations*.
 - e.g.* four fold degenerate codons tolerate any point mutation on the third position meaning that an error (silent mutation) would not affect the protein.
- Since 2004, 40 non-natural amino-acids has been added into protein, creating a uniquely decodable genetic code in order to be used as a tool in exploring proteins structure and functions or to create novel enhanced proteins [44], [42].

Bioinformatics, the “computational branch of molecular biology” or “in silico biology [10], started 40 years ago when the early computers were used to analyse molecular segments as texts, has tremendously evolved since than, being at the centre of the greatest development in biology and other connected fields: deciphering of human genome, new biotechnologies, personalized medicine, bioarcheology, anthropology, evolution and human migration.

3.4 Coding Efficiency: Compression Ratio

The correspondence between s_i and c_i achieved by coding can be accomplished in various ways using the alphabet X . To choose a code from more possible ones can be done if an optimisation criterion is used. The optimisation criterion for a noiseless channel is the transmission cost, which is desired to be minimal, thus a minimum number of symbols entering the channel. For information storage systems, the optimisation criterion is the storage space, which as in the transmissions case, is desired to be minimal.

We define the *length of a word* (l_i) as the number of letters that compose a codeword, in order to be able to make remarks upon this criterion. Each letter is assumed to have the same duration, supposed 1.

Average length of the codewords (\bar{l}), is the average value of these lengths:

$$\bar{l} = \sum_{i=1}^M p_i l_i \quad (3.43)$$

Taking into account that, by encoding, to every message s_i a correspondent codeword c_i is made, we have:

$$p(s_i) = p(c_i) = p_i, \quad i = \overline{1, M} \quad (3.44)$$

The transmission cost is proportional to \bar{l} ; therefore it is desirable to get \bar{l}_{\min} by encoding.

The question that raises now is: $\bar{l}_{\min} = ?$

Under the assumption of a DMS, the average information quantity per symbol is $H(S)$ and it is equal to the average information quantity per codeword c_i ; hence:

$$H(S) = H(C) = - \sum_{i=1}^M p_i \log_2 p_i \quad (3.45)$$

The average quantity of information per each code letter is $H(X)$:

$$H(X) = - \sum_{j=1}^m p(x_j) \log_2 p(x_j) = - \sum_{j=1}^m p_j \log_2 p_j \quad (3.46)$$

It follows

$$H(S) = \bar{l} \cdot H(X) \quad (3.47)$$

known as *entropic or lossless compression relation*.

Finally, we get:

$$\bar{l} = \frac{H(S)}{H(X)} \quad (3.48)$$

The minimal value for \bar{l} is obtained for $H(X) = H_{\max}(X) = D(X)$, meaning the use of the alphabet X with equal probability: $p_{i_0} = \frac{1}{m}, \forall i = \overline{1, m}$, which leads to:

$$\bar{l}_{\min} = \frac{H(S)}{\log_2 m} \quad (3.49)$$

Remark

1. The same condition $p_{i_0} = \frac{1}{m}, \forall i = \overline{1, m}$, known as statistical adaptation of the source to the channel (see relation 3.4) is obtained, this time from *coding theory point of view*.
2. Relation (3.49) is valid only when the alphabet X is memoryless; however, this is not always the case [6].
3. Equations (3.47) and (3.49) show that encoding a DMS, its entropy is preserved, the source being changed into another one, X , with maximum entropy. This justifies the denomination of this type of compression as entropic or loss-less compression.

Coding efficiency (η) is defined as:

$$\eta = \frac{\bar{l}_{\min}}{\bar{l}} = \frac{H(S)}{\bar{l} \log_2 m} \quad (3.50)$$

Another measure, more often used in compression techniques that also emphasises the coding efficiency, is the *compression ratio* R_C defined as the ratio between the length of the codewords in uniform encoding l_u and the average length of the codewords obtained by encoding:

$$R_C = \frac{l_u}{\bar{l}} \quad (3.51)$$

The codes for which $\bar{l} = \bar{l}_{\min}$ are named *absolute optimal codes* (AOC). For AOC we have

$$H_{\max}(X) = D(X) = \log_2 m$$

so the symbols x_j have equal probabilities: $p(x_j) = 1/m, \forall j = \overline{1, m}$.

Taking into account that $p(s_i) = p(c_i)$, it follows that for a DMS (symbols x_j are assumed independent) we have:

$$p(c_i) = \left(\frac{1}{m}\right)^{l_i} \quad (3.52)$$

Using the following equalities

$$\sum_{i=1}^M p(s_i) = \sum_{i=1}^M p(c_i) = \sum_{i=1}^M \left(\frac{1}{m}\right)^{l_i} = \sum_{i=1}^M m^{-l_i} = 1$$

we also get:

$$\sum_{i=1}^M m^{-l_i} = 1 \quad (3.53)$$

This relation is valid for AOC and it shows the connection that must exist between the lengths l_i and the code alphabet m for absolute optimal codes.

Remark

If (3.53) is fulfilled for a particular code, it does not necessarily mean that the code is an absolute optimal one; however, it states that for a given alphabet m , for words of lengths l_i that fulfils (3.53), an AOC can always be built.

Example 3.3

Source S is encoded in two different ways as follows:

$$S: \begin{pmatrix} s_1 & s_2 & s_3 & s_4 \\ 0,5 & 0,25 & 0,125 & 0,125 \end{pmatrix}$$

$$C_1: \begin{pmatrix} 111 & 110 & 10 & 0 \\ l_{i_1} & 3 & 3 & 2 & 1 \end{pmatrix} \quad C_2: \begin{pmatrix} 0 & 10 & 110 & 111 \\ l_{i_2} & 1 & 2 & 3 & 3 \end{pmatrix}$$

One must notice that (3.53) is fulfilled for both C_1 and C_2 :

$$2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = 1,$$

but even so, only C_2 is an AOC

$$\eta_1 = \frac{H(S)}{\log 2} = \frac{1,75}{2,6} < 1$$

$$\eta_2 = 1$$

3.5 Existence Theorem for Instantaneous and Uniquely Decodable Codes (Kraft and McMillan Inequalities)

Theorem T_1

The necessary and sufficient condition for instantaneous codes existence is:

$$\sum_{i=1}^M m^{-l_i} \leq 1 \quad (3.54)$$

This relationship is also known as *Kraft inequality*.

Proof

a) necessity:

We will show that (3.54) is fulfilled for an instantaneous code of base m with words of length $l_1 \leq l_2 \leq \dots \leq l_M$.

The code can be drawn as a graph of order m and size l_M . Being an instantaneous code, it means that no codeword must be a prefix of another codeword, so that once a codeword is identified on the tree, it is impossible that another codeword exist on the incident branches representing the identified codeword.

This mean that for every codeword of length l_i , $m^{l_M-l_i}$ ends of the tree are excluded.

For the given code, the total number of excluded ends is:

$$\sum_{i=1}^M m^{l_M-l_i}$$

and taking into account that the number of ends for a tree of size l_M is m^{l_M} , one of the following two inequalities must be true:

$$\sum_{i=1}^M m^{l_M-l_i} \leq m^{l_M}, \text{ or:}$$

$$\sum_{i=1}^M m^{-l_i} \leq 1$$

b) sufficiency:

It will be shown that using an m base alphabet and taking lengths l_i so that $l_1 \leq l_2 \leq \dots \leq l_M$, an instantaneous code can be built, under the assumption that Kraft inequality holds.

Let us imagine an m order graph of size l_M . An end of this graph is considered to be a codeword of length l_1 , and thus $m^{l_M-l_1}$ ends eliminated. But taking into consideration that $\sum_{i=1}^M m^{l_M-l_i} \leq m^{l_M}$, at least one end that can be considered a codeword of length l_2 still remains; therefore we have:

$$m^{l_M-l_1} + m^{l_M-l_2} \leq m^{l_M}$$

The algorithm is applied for codeword of lengths l_3, l_4, \dots

Theorem T₂

Existence theorem for uniquely decodable codes, *McMillan theorem*: Kraft inequality, (3.54), is a necessary and sufficient condition for UDC existence.

The proof of this theorem is similar to the one made for Kraft theorem [33].

Theorems T₁ and T₂, given by (3.54) are existence theorems; they show that for a given alphabet m , with lengths l_i , obeying (3.54), instantaneous or uniquely decodable codes can be built: notice that these theorems do not give algorithms for IC or UDC codes.

3.6 Shannon First Theorem (1948)

As shown in 3.4 (AOC), for the special case when source messages have certain particular values as:

$$p_i = p(s_i) = p(c_i) = m^{-l_i} \quad (3.55)$$

the coding efficiency is maximum ($\eta=1$). From (3.55) it follows that:

$$l_i = -\frac{\log_2 p_i}{\log_2 m} \in Z \text{ (integer set)} \quad (3.56)$$

For a source with a random set of probabilities, the ratio given by (3.56) is not an integer and therefore will be rounded upwards until obtaining the closest superior integer:

$$\begin{aligned} -\frac{\log_2 p_i}{\log_2 m} \leq l_i < -\frac{\log_2 p_i}{\log_2 m} + 1 \Big| \cdot p_i & \quad (3.57) \\ -\frac{p_i \log_2 p_i}{\log_2 m} \leq l_i p_i < -\frac{p_i \log_2 p_i}{\log_2 m} + p_i \Big| \sum_{i=1}^M & \\ -\sum_{i=1}^M \frac{p_i \log_2 p_i}{\log_2 m} \leq \sum_{i=1}^M l_i p_i \leq -\sum_{i=1}^M \frac{p_i \log_2 p_i}{\log_2 m} + \sum_{i=1}^M p_i & \\ \frac{H(S)}{\log_2 m} \leq \bar{l} < \frac{H(S)}{\log_2 m} + 1 & \quad (3.58) \end{aligned}$$

Relationship (3.58) is valid for any DMS, and in particular for the n-th extension of the source S, for which $H(S^n) = nH(S)$; it follows that:

$$\frac{nH(S)}{\log_2 m} \leq \bar{l}_n < \frac{nH(S)}{\log_2 m} + 1,$$

where \bar{l}_n is the average length obtained encoding the source extension S^n .

$$\frac{H(S)}{\log_2 m} \leq \frac{\bar{l}_n}{n} = \bar{l} < \frac{H(S)}{\log_2 m} + \frac{1}{n} \quad (3.59)$$

For $n \rightarrow \infty$, we obtain:

$$\lim_{n \rightarrow \infty} \frac{\bar{l}_n}{n} = \bar{l} = \frac{H(S)}{\log_2 m} = \bar{l}_{\min} \quad (3.60)$$

Relation (3.60) represents *Shannon first theorem or noiseless channels coding theorem*; it shows that when coding on groups of n symbols, an absolute optimal encoding can be achieved for any source S, under the assumption: $n \rightarrow \infty$. This

was expected due to the fact that when coding on groups of symbols, rounding upwards refers to the entire group and thus the rounding corresponding to only one symbol is smaller than the one obtained when coding symbol by symbol, so we will approach as much as possible to \bar{l}_{\min} .

Example 3.4

Be $S: \begin{pmatrix} s_1 & s_2 \\ 0.7 & 0.3 \end{pmatrix}$ encoded with $C_a = \{0;1\}$.

It follows that:

$$\bar{l}_{\min} = H(S) = 0,88 \text{ bits/symbol}$$

$$\bar{l}_a = 1, \eta_a = 0,88$$

The 2nd order extension of the source is:

$$S^2: \begin{pmatrix} s_1s_1 & s_1s_2 & s_2s_1 & s_2s_2 \\ 0.49 & 0.21 & 0.21 & 0.09 \end{pmatrix} \text{ encoded with}$$

$$C_b: (1 \ 01 \ 000 \ 001)$$

In this case we get: $\bar{l}_b = 1,8$

Relationship (3.21) becomes: $\frac{\bar{l}_n}{n} = \frac{\bar{l}_b}{n} = \frac{1,8}{2} = 0,9 < \bar{l}_a$; one may notice a decrease of the average length per symbol when coding on groups of two symbols.

We invite the reader to check coding efficiency improvements when coding on groups of $n=3$ symbols.

3.7 Lossless Compression Algorithms

Shannon first theorem shows that for any source S , an absolute optimal coding is possible, if performed on the n -th order extension, with $n \rightarrow \infty$.

In practice n is finite. The algorithms called *optimal algorithms* are ensuring an encoding for which $\bar{l} \rightarrow \bar{l}_{\min}$.

The *basic idea* when dealing with optimal algorithms is to associate to high probabilities p_i , short codewords (small l_i), and conversely fulfilling, obviously, Kraft inequality.

Remark

This basic idea for optimal codes is not new, a relevant example being Morse code, dating from 1837.

3.7.1 Shannon-Fano Binary Coding

Shortly after stating its first theorem, Shannon gave the first optimal coding algorithm, in 1949; R.M. Fano, simultaneously, proposed the same algorithm and this is why the algorithm is known as *Shannon – Fano algorithm*.

Although efficient, this algorithm was proved not to be the best for any kind of source; Huffman algorithm, invented later, proved to be the best one, and thus optimal.

The main steps of Shannon-Fano algorithm are:

1. Put the set $S = \{s_i\}, i = \overline{1, M}$ in decreasing order of probabilities.
2. Partition S into two subsets (S_0, S_1) of equal or almost equal probabilities $P(S_0) \cong P(S_1) \cong 1/2$.
3. Repeat step 2 for subsets S_0 and S_1 and get new subsets S_{00}, S_{01} and S_{10}, S_{11} respectively, of probabilities as close to $1/4$. Repeat until each subset contains only one message s_i .
4. Assign 0 to S_0 and 1 to S_1 or vice versa such that the codewords assigned to the symbols corresponding to S_0 begin with 0 and the codewords assigned to the symbols corresponding to S_1 begin with 1.
5. Repeat step 4 until the last subsets that contain one message.

Example 3.5

Be a DMS given by the probability mass function:

$$S: \begin{pmatrix} s_1 & s_2 & s_3 & s_4 \\ 1/2 & 1/4 & 1/8 & 1/8 \end{pmatrix}$$

Determine:

- a) binary Shannon-Fano code corresponding to the source.
- b) the code tree.
- c) encoding efficiency and the compression ratio.
- d) statistical adaptation of the source to the channel, made by encoding.

Solution

a)

s_i	p_i	Partitions		c_i	l_i	l_{i0}	l_{i1}	
s_1	1/2	0		0	1	1	0	
s_2	1/4	1	0	10	2	1	1	
s_3	1/8		1	0	110	3	1	2
s_4	1/8			1	111	3	0	3

b) The graph corresponding to the code is given in Fig. 3.11

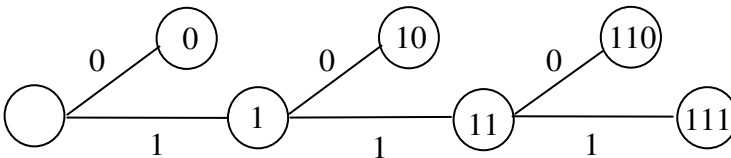


Fig. 3.11 The coding tree of Example 3.5

One may notice that an instantaneous code is obtained.

$$\begin{aligned} \text{c) } \bar{l} &= \sum_{i=1}^M p_i l_i = 1/2 \times 1 + 1/4 \times 2 + 1/8 \times 3 + 1/8 \times 3 = 1,75 \\ \bar{l}_{\min} &= \frac{H(S)}{\log_2 m} = H(S) = -\sum_{i=1}^M p_i \log_2 p_i = 1,75 \\ \eta_c &= \frac{\bar{l}_{\min}}{\bar{l}} = \frac{1,75}{1,75} = 1 = 100\%, \end{aligned}$$

therefore the code is absolute optimal; this could have been easily noticed even from the beginning as the source probabilities lead to integer lengths l_i :

$$\begin{aligned} -\log_2 p_1 &= -\log_2 1/2 = 1 = l_1 \\ -\log_2 p_2 &= -\log_2 1/4 = 2 = l_2 \\ -\log_2 p_3 &= -\log_2 1/8 = 3 = l_3 \\ -\log_2 p_4 &= -\log_2 1/8 = 3 = l_4 \end{aligned}$$

$$R_C = \frac{l_u}{l}$$

where l_u is determined from:

$$m^{l_u} \geq M \quad (3.61)$$

It follows that:

$$l_u = \frac{\text{ld}M}{\text{ld}m} \quad (3.62)$$

We get $l_u = 2$, therefore $R_C = 2 / 1,75 = 1,14$

d) Statistical adaptation of the source to the channel involves the determination of the probabilities corresponding to the code alphabet, therefore of $p(0)$ and $p(1)$, which must be as close as possible; for an AOC we must have $p(0)=p(1)=1/2$. The two probabilities are computed using the equations:

$$p(1) = \frac{N_0}{N_0 + N_1} = \frac{\sum_{i=1}^M p_i l_{i0}}{\sum_{i=1}^M p_i l_{i0} + \sum_{i=1}^M p_i l_{i1}} \quad (3.63)$$

$$p(1) = \frac{N_1}{N_0 + N_1} = \frac{\sum_{i=1}^M p_i l_{i1}}{\sum_{i=1}^M p_i l_{i0} + \sum_{i=1}^M p_i l_{i1}} \quad (3.64)$$

where N_0 and N_1 are the average number of zeros and ones used in source coding.

We obtain $p(0)=p(1)=1/2$ and therefore the statistical adaptation of the source to the channel was accomplished through encoding point of view; this means that

such a source will use the binary symmetric channel at its full capacity (information theory point of view).

3.7.2 Huffman Algorithms

Binary static Huffman coding (1952)

Huffman algorithm (1952) is an optimal one, meaning that no other algorithm ensures a shorter average length. There are cases when other algorithms can provide an average length equal to Huffman one, but never smaller.

The steps of this algorithm are:

1. Put the source $S=\{s_i\}$ messages in decreasing order of probabilities.
2. Combine the two most probable messages into a reduced message $r_1 = s_{M-1} \cup s_M$ having the probability:
 $p(r_1) = p(s_{M-1}) + p(s_M)$.
3. Include the message r_1 into the remaining messages set and put this set in decreasing order of probabilities, obtaining the ordered stream R_1 .
4. Repeat the reduction algorithm until the last ordered stream R_k contains only two messages $R_k=\{r_{k-1}, r_k\}$.
5. The codewords corresponding to each message are obtained as follows:
 - assign '0' to the symbol r_{k-1} and '1' to r_k
 - assign '0's and '1's to each restriction until singular messages (s_i) are obtained.

Example 3.6

- a) Encode using Huffman binary algorithm the source S:

$$S: \begin{pmatrix} s_1 & s_2 & s_3 & s_4 & s_5 & s_6 \\ 0.3 & 0.25 & 0.15 & 0.15 & 0.10 & 0.05 \end{pmatrix}$$

- b) Calculate the compression efficiency and the compression ratio
 c) Show the fulfilment of lossless compression relation. Discussion.

Solution

- a)

				Restrictions			
s_i	p_i	c_i	l_i	R_1	R_2	R_3	R_4
s_1	0,3	00	2	0,3	0,3	0,4	0,6 0
s_2	0,25	10	2	0,25	0,3	0,3 00	0,4 1
s_3	0,15	11	2	0,15	0,25 10	0,3 01	
s_4	0,15	010	3	0,15 010	0,15 11		
s_5	0,10	0110	4	0,15 011			
s_6	0,05	0111	4				

$$\begin{aligned} \text{b) } \bar{l} &= \sum_{i=1}^6 p_i l_i = 2.45 \\ H(S) &= -\sum_{i=1}^6 p_i \log_2 p_i = 2.39 \\ \eta &= \frac{2.39}{2.45} = 0.98 = 98\% \\ R_c &= \frac{l_u}{\bar{l}} = \frac{3}{2.45} = 1.225 \end{aligned}$$

- c) Using (3.63) and (3.64), we obtain: $p(0) \approx 0.6$, $p(1) \approx 0.4$ and $H(X) = 0.9785$. The lossless compression relation (3.47) becomes:

$$2.39 = H(S) = \bar{l} \cdot H(X) = 2.45 \cdot 0.9785 = 2.3973.$$

Discussion:

Relation (3.47) was given under the assumption of S and X memoryless sources. Encoding introduces a slight memory, meaning that X is not completely memoryless.

Non-binary static Huffman encoding ($m > 2$)

Huffman also presented the general case of its algorithm for a channel with an alphabet containing more than two symbols ($m > 2$). Unlike for binary coding, when coding in a larger m base, each reduction will be formed from m symbols. In this case, after the first reduction we obtain a new source having $M - (m - 1)$ symbols, and after the last reduction (of order k) it has $M - k(m - 1)$ symbols. It must be underlined, that for an accurate encoding, the last reduction must contain exactly m elements, therefore:

$$M - k(m - 1) = m \tag{3.65}$$

When the number of source messages does not allow an accurate encoding in the sense given by (3.65), this number is increased, the new messages having zero probabilities and therefore not affecting the initial source.

Example 3.7

Encode the source from Example 3.6 using a ternary alphabet ($m = 3$) and determine the efficiency and the compression ratio.

Solution

The number of messages must check (3.65).

$$M = k(m - 1) + m = k(3 - 1) + 3 = 2 \times 2 + 3 = 7, k \in \mathbb{Z}$$

Note that a supplementary symbol s_7 of zero probability must be added.

s_i	p_i	c_i	R_1	R_2
s_1	0,3	1	0,3	0,45 0
s_2	0,25	2	0,25	0,3 1
s_3	0,15	00	0,15 00	0,25 2
s_4	0,15	01	0,15 01	
s_5	0,10	020	0,15 02	
s_6	0,05	021		
s_7	0,0	022		

$$\bar{l} = 0.3 \times 1 + 0.25 \times 1 + 0.15 \times 2 + 0.10 \times 3 + 0.05 \times 3 = 1.85$$

$$\bar{l}_{\min} = \frac{H(S)}{\log_2 m} = \frac{2.39}{1.584} = 1.5$$

$$\eta = \frac{1.5}{1.85} = 0.81 = 81\%$$

$$R_C = \frac{l_u}{1} \text{ where } l_u \text{ is } l_u = \frac{\log_2 M}{\log_2 m} = \frac{\log_2 6}{\log_2 3} = \frac{2.584}{1.584} = 1.63$$

therefore we choose $l_u=2$.

Replacing $l_u=2$ in the compression ratio formula, we get:

$$R_C = \frac{2}{1.5} = 1.33$$

Remarks concerning static Huffman algorithm

- The obtained codes are not unique. Interchanging 0 with 1 (for binary codes) we obtain a complementary code. Similarly, for messages with equal probabilities, assigning the codewords is arbitrarily. However, even though the code is not unique, all the codes obtained with the same algorithm provide same average length, therefore same efficiency and compression ratio.
- The last m codewords length is the same, if m -ary alphabet is used.
- The codes, obtained using the described algorithms, are instantaneous; therefore no codeword is prefix for another codeword.
- Symbol by symbol Huffman coding, in the case when the highest probability is close to 1, has as result a code efficiency decrease. This drawback can be overcome using Huffman coding for streams of symbols (see Shannon’s first theorem for $n>1$). In this case the information is divided in fixed length blocks that are then encoded.
- Huffman coding is frequently used as the final processing (back-end coder), in a series of compression schemes.

Multi-group Huffman coding [41]

This type of coding is used for sources that generate streams of characters belonging to disjunctive classes, for example sources generating burst letters followed by numbers and then by blank spaces. When designing such an encoder a tree is set up for each class of symbols. A failure symbol is then added to each tree. Each character is coded taking into consideration the current Huffman tree. In the case when the character can be found in that particular tree, its corresponding code is transmitted; otherwise the code corresponding to the failure character is transmitted pointing to another tree.

Advanced applications of this type can be found in database system drivers.

Dynamic Huffman coding (1978)

All Huffman-type algorithms that have been presented so far bear one major disadvantage: all require the source statistics (*static algorithms*). This drawback can be overcome using an adaptive (dynamic) algorithm. R.G. Gallager [14] presented in 1978 three new theorems on dynamic Huffman codes. D.E. Knuth [24], using Gallager theorems, presented a Huffman algorithm with the capability of dynamically changing its own tree.

The *basic idea* of dynamic Huffman algorithm is to use, for coding the symbol s_{i+1} , a coding tree (a coding dictionary set up as a binary tree) constructed using the first i symbols of the message. After transmitting the symbol s_{i+1} the coding tree must be revised in order to code the next symbol s_{i+2} . There are more dynamic Huffman versions (FGK, Δ) [2] each of them using a different method for setting up the tree.

In what follows we will introduce the pseudo-code corresponding to the dynamic Huffman compression algorithm.

The general compression procedure for dynamic Huffman coding has the following pseudo-code:

- initialise the coding tree with a root node;
- transmit the first symbol as it is (using, for example, its ASCII code);
- add 2 leaves to the root node (a left leaf, empty leaf of weight 0, a right leaf of weight 1, which contains the current symbol)

while (end of message)

```

{
  - read a message symbol;
  if (the letter already belongs to the tree)
    - transmit its code from the tree;
  else
    {
      - transmit the empty leaf code;
      - transmit the ASCII code of the letter;
    }
  update tree;
}

```

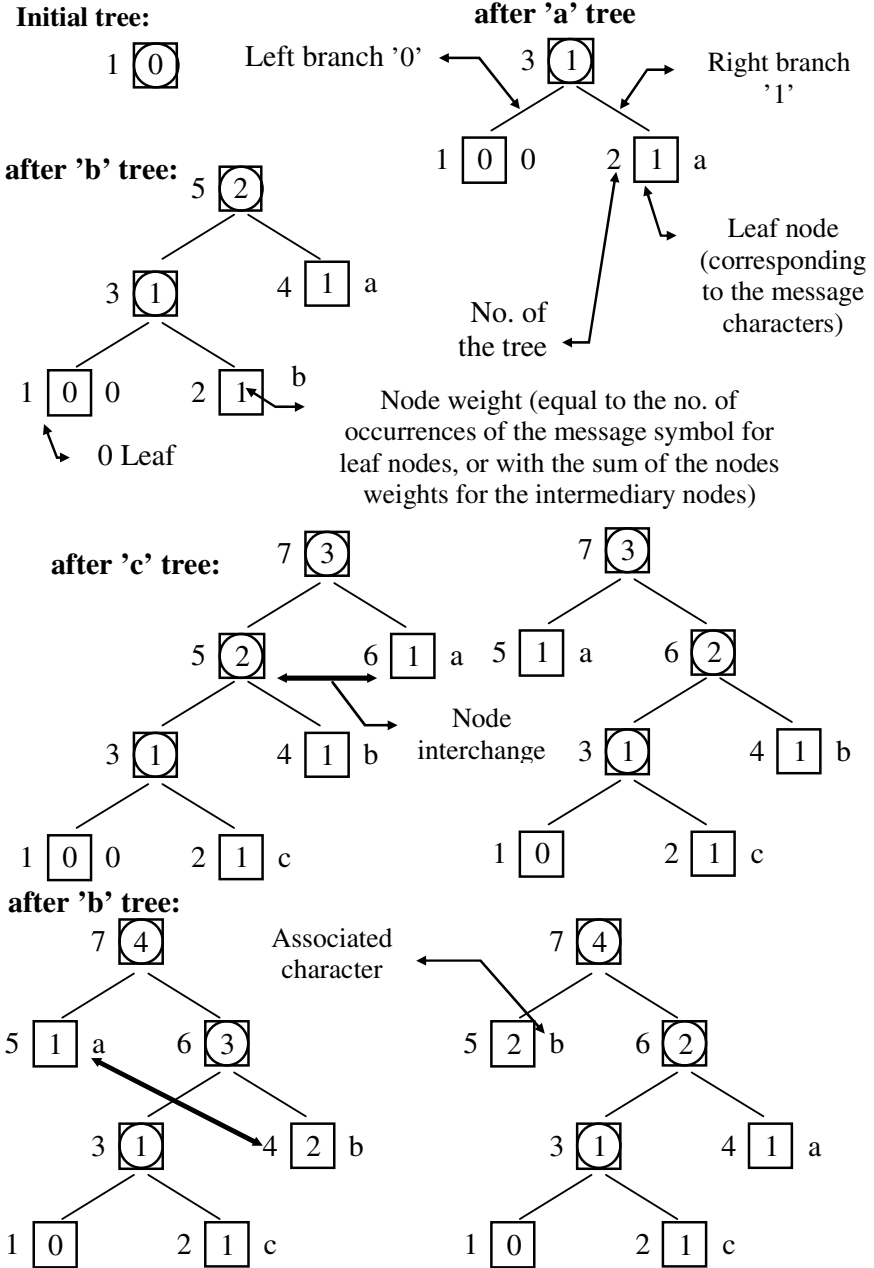


Fig. 3.12 Evolution of dynamic Huffman FGK tree for the message 'abcbb'

Remark

The only difference between the numerous versions of dynamic coding is the way the coding tree is updated.

One version of updating the tree is the FGK (Faller, Gallager, and Knuth) algorithm. The basic idea for this method is the minimisation of

$$\sum_j w_j l_j \quad (\text{sum equivalent to the bit-length of the coded message}),$$

where w_j represents the weight of the leaf j associated to the symbol j (the number of appearances of the j symbol) and l_j is the code length associated to that leaf.

This method is used in the MNP5 (Microcom Network Protocol 5) data compression standard for modems.

The pseudo-code corresponding to the FGK updating procedure is:

```

-  $q$  = the leaf corresponding to the symbol  $s_i$  if this symbol can be found in
the tree, or an empty leaf if it doesn't belong to the tree;
if ( $q$  is a 0 leaf)
{
- substitute the 0 leaf with a parent node with two 0 leafs sons num-
bered: 1 the left son, 2 the right son, 3 the parent;
- increment the order of the other nodes;
-  $q$  = the new created right son;
}
while ( $q$  is not a root node)
{
- increment the weigh of the node  $q$ ;
- replace  $q$  with the highest ranked and smallest weighted node;
-  $q$  =  $q$  node parent;
}

```

Huffman coding for Markov sources

Until now Huffman coding was applied only for memoryless discrete sources. A question rises: how to apply this algorithm for memory sources? The answer to this question can be given in two ways:

- the probability mass function of the stationary state P^* is encoded obtaining a unique code.
- the Markov source is encoded for all M states from where the source can evolve, therefore determining M codes.

Let us now code the Markov source from Example 2.7 for which we have:

$$M = \begin{bmatrix} 0,6 & 0,2 & 0,2 \\ 0,3 & 0,4 & 0,3 \\ 0,2 & 0,2 & 0,6 \end{bmatrix}$$

and the stationary state probability mass function is:

$$P^* = [3/8 \quad 1/4 \quad 3/8]$$

a. Stationary state Huffman coding:

s_i	p_i	c_i
s_1	$3/8$	1
s_2	$3/8$	00
s_3	$1/4$	01

The average length of the codewords is:

$$\bar{l} = 3/8 \times 1 + 3/8 \times 2 + 1/4 \times 2 = 1.62$$

b₁. Huffman coding, the source starting from state s_1 :

s_i	p_i	c_i
s_1	0,6	0
s_2	0,2	10
s_3	0,2	11

$$\bar{l}_1 = 0.6 \times 1 + 0.2 \times 2 + 0.2 \times 2 = 1.4$$

b₂. Huffman coding, the source starting from state s_2 :

s_i	p_i	c_i
s_1	0,4	1
s_2	0,3	00
s_3	0,3	01

$$\bar{l}_2 = 0.4 \times 1 + 0.3 \times 2 + 0.3 \times 2 = 1.6$$

b₃. Huffman coding, the source starting from state s_3 :

s_i	p_i	c_i
s_1	0,6	0
s_2	0,2	10
s_3	0,2	11

$$\bar{l}_3 = 0.6 \times 1 + 0.2 \times 2 + 0.2 \times 2 = 1.4$$

The average length of Huffman code applied to this source will be determined as the average value of the lengths determined previously:

$$\bar{l}_M = p_1^* \times \bar{l}_2 + p_2^* \times \bar{l}_2 + p_3^* \times \bar{l}_3 = 3/8 \times 1.4 + 1/4 \times 1.6 + 3/8 \times 1.4 = 1.45$$

Remark

If the number of states is high, the average length decreases slower and slower and the number of distinct codes increases so that coding the stationary state becomes more practical [16].

Conclusions concerning Huffman coding

- This algorithm is probably one of the most used compression algorithms.
- It is a rigorous, simple and elegant algorithm appreciated both by theorists and practitioners. Despite its “age”, experimental studies in which it is used still appear.
- The algorithm can be applied in three distinct versions:
 - static: knowledge of the source statistics $P(S)$ is compulsory
 - quasi-dynamic (half-adaptive): as a first step finds the source PMF and then uses static encoding
 - dynamic (adaptive): it does not require the knowledge of source statistics in advance as the tree is constructed with each read symbol
- The use of the static version of this compression algorithm in transmissions is limited to known sources for which the PMF is known: $P(S)$ (e.g. for compressed transmissions of texts written in different languages, the encoder-decoder can be designed according to a well known statistic $P(S)$).
- Huffman algorithm – being optimal at message level – was thought to be the best compression algorithm. However, in practical applications, this statement has limitations; it is forgotten that it was defined in restrictive conditions:
 - memoryless source
 - symbol by symbol encoding

Most of the real sources are not memoryless and the encoding can be done on groups (using Shannon first theorem) therefore it is possible to achieve better compression ratios R_C than the ones given by Huffman algorithms, although the former compressions are not optimal.

Between 1952 and 1980, after Huffman work publication and computer spreading, data compression had a major development both theoretically and practically. Some data compression techniques occurred and were named, although not always accurately, *ad-hoc compression* techniques.

Many researches in this field have focused on coding source extensions for which each symbol corresponds to a stream of symbols of the initial source.

3.7.3 Run Length Coding (RLC)

The *basic idea* of run length coding consists in replacing a stream of repeating characters with a compressed stream containing fewer characters. This type of coding is especially used for memory sources (repetition means memory).

Principle of run length coding:

- successive samples are analysed and if a number of r successive samples differ with less than α quanta, it is said that we have a r -length step (α is named the compression algorithm aperture; identical samples implies $\alpha=0$)
- for transmissions or storage, it is enough to provide only the first sample amplitude and the step length ‘ r ’, in binary.

Huffman coding of steps (offers the best compression):

- the first sample from every step is stored together with the codeword which represents the step length (r_i)
- the data statistics $p(r_i)$, necessary for encoding, is obtained experimentally.

Example 3.8 [38]

Consider a fax transmission (black and white image). Taking into consideration streams of the same type (black or white) with length r_i , we can determine their PMF, for the whole image.

In this case, the source alphabet will be:

$$A = \{r_1, r_2, \dots, r_M\},$$

where r_i indicate streams of the same length r_i (black or white).

The corresponding PMF, obtained as a result of a statistical analysis of the image, is given by:

$$P(r_i) = [p_i], \quad i = \overline{1, m}, \quad \sum_{i=1}^M p_i = 1$$

Assuming independent steps, the average information quantity per step is:

$$H(A) = -\sum_{i=1}^M p_i \log_2 p_i \quad [\text{bits/step}] \quad (3.66)$$

These steps can be optimally encoded using the binary Huffman algorithm; it results, for A , the following average length:

$$H(A) \leq \bar{l} < H(A) + 1 \quad (3.67)$$

Dividing (3.67) by the average number of samples per step (\bar{r}) we get:

$$\frac{H(A)}{\bar{r}} \leq \frac{\bar{l}}{\bar{r}} < \frac{H(A) + 1}{\bar{r}} \quad (3.68)$$

It follows that:

$$\frac{H(A)}{\bar{r}} \leq \frac{\bar{l}}{\bar{r}} = \overset{\bullet}{D}_b < \frac{H(A) + 1}{\bar{r}} \quad (3.69)$$

where $\overset{\bullet}{D}_b$ is the binary rate [bits/pixel (sample)].

$$\overset{\bullet}{D}_b = \frac{\bar{l}}{\bar{r}} \quad (3.70)$$

For a typical fax image, corresponding to a weather forecast image [38], for this type of compression we get: $\frac{H(A)}{\bar{r}} \cong 0.2$ [bits/pixel] compared to '1' obtained without compression.

Remark

A better modelling of this information source can be made using a first order Markov model, in which case the compression is even better.

Example 3.9 [38]

The fax standard, proposed by the 3rd research group from CCITT, (Hunter and Robinson, 1980) has a resolution of 1728 [samples/line] for A4 documents so the maximum number of codewords/line is 1728. Due to this great number of codewords the ‘classic’ Huffman code becomes useless, therefore a *modified Huffman code* has been introduced; each step with length $r_i > 63$ is divided in 2 steps: one having the value $N \times 64$ (N is an integer) - “the root word”, and the other one - “the terminal part”, made of steps with values between 0 and 63. The black and white streams are encoded separately as they are independent, obtaining a lower rate (and thus a better compression). Table 3.12 shows the modified Huffman code used in facsimiles.

Table 3.12 Modified Huffman code used in fax

Codewords					
r_i	White	Black	r_i	White	Black
0	00110101	0000110111	32	00011011	000001101010
1	000111	010	33	00010010	000001101011
2	0111	11	34	00010011	000011010010
3	1000	10	35	00010100	000011010011
4	1011	011	36	00010101	000011010100
5	1100	0011	37	00010110	000011010101
6	1110	0010	38	00010111	000011010110
7	1111	00011	39	00101000	000011010111
8	10011	000101	40	00101001	000001101100
9	10100	000100	41	00101010	000001101101
10	00111	0000100	42	00101011	000011011010
11	01000	0000101	43	00101100	000011011011
12	001000	0000111	44	00101101	000001010100
13	000011	00000100	45	00000100	000001010101
14	110100	00000111	46	00000101	000001010110
15	110101	000011000	47	00001010	000001010111
16	101010	0000010111	48	00001011	000001100100
17	101011	000011000	49	01010010	000001100101
18	0100111	0000001000	50	01010011	000001010010
19	0001100	00001100111	51	01010100	000001010011
20	0001000	00001101000	52	01010101	000000100100
21	0010111	00001101100	53	00100100	000000110111
22	0000011	000001101111	54	00100101	000000111000
23	0000100	00000101000	55	01011000	000000100111
24	0101000	00000010111	56	01011001	000000101000
25	0101011	00000011000	57	01011010	000001011000

Table 3.12 (continued)

26	0010011	000011001010	58	01011011	000001011001
27	0100100	000011001011	59	01001010	000000101011
28	001000	000011001100	60	01001011	000000101100
29	00000010	000011001101	61	00110010	000001011010
30	00000011	000001101000	62	00110011	000001100110
31	00011010	000001101001	63	00110100	000001100111
“Root” codewords					
r_i	White	Black	r_i	White	Black
64	11011	0000001111	960	011010100	0000001110011
128	10010	000011001000	1024	011010101	0000001110100
192	0010111	000011001001	1088	011010110	0000001110101
256	0110111	000001011011	1152	011010111	0000001110110
320	00110110	000000110011	1216	011011000	0000001110111
384	00110111	000000110100	1280	011011001	0000001010010
448	01100100	000000110101	1344	011011010	0000001010011
512	01100101	0000001101100	1408	011011011	0000001010100
576	01101000	0000001101101	1472	010011000	0000001010101
640	01100111	0000001001010	1536	010011001	0000001011010
704	011001100	0000001001011	1600	010011010	0000001011011
768	011001101	0000001001100	1664	011000	0000001100100
832	011010010	0000001001101	1728	010011011	0000001100101
896	011010011	0000001110010	EOL	000000000001	000000000001

Remark

If the compression is made using variable-length coding (Huffman coding), removing the redundancy, the transmission becomes vulnerable to errors. In general (but not in all the cases) an error occurred in transmission will spread, leading to synchronisation loss and, implicitly, to an incorrect decoding.

5W	2B	2W	3B	25W	
1 1 0 0	1 1	0 1 1 1	1 0	0 1 0 1 0 1 1	-a
5W	3B	2W	3B	25W	
1 1 0 0	1 0	0 1 1 1	1 0	0 1 0 1 0 1 1	-b
5W	1B	7W	6B	4W	
1 1 0 0	0 1 0	1 1 1 1	0 0 1 0	1 0 1 1	-c

Fig. 3.13 The effect of one error on the modified Huffman code: a) transmitted sequence, b) and c) received sequence affected by one error

Coding using a limited set of step lengths

Another way of compressing identical streams, although not optimal as in Huffman coding, can be achieved using a limited set of codewords lengths corresponding to each step.

Example 3.10 [38]

- Images in the visible and infrared spectrum taken by the weather forecast satellites can be classified in three classes, according to the homogeneous areas they represent: the first class (water and land surfaces) and two classes of clouds (C_1 , C_2). The homogenous regions are suitable for run length coding before transmission. Two sets of lengths, of 8 and 16 bits are used.
- If the steps length $r_i < 64 = 2^6$, the images are coded using a word of $l_1 = 8$ bits, the first two bits indicating the region:

$b_1 b_2 = 00$ indicates water + land

$01 \rightarrow C_1$

$10 \rightarrow C_2$

b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
indicate the region		6 bits for uniform coding of r_i steps					

- If the steps length $r_i \geq 64$, the extended code is used, therefore $l_2 = 16$ bits; the 16 bits are used as shown below:

b_1	b_2	b_3	b_4	b_5	b_6	b_{16}
11 extended code		indicate the region				12 bits for uniformly coding the steps r_i	

It has been determined that, for this application, approximately 94% of streams have $r_i < 64$, therefore 8 bit coding is used. Average length of the codewords is 8.5 bits/stream. For regular weather images a compression ratio of $R_c \cong 4$ was obtained.

Uniform step coding

The third type of RLC, the poorest in quality, is using uniform step coding. For a given source, be r_1, \dots, r_M the steps lengths, where r_M is the maximum length. All these steps will be uniformly encoded, the codewords length being determined by: $l = \log_2 r_M$.

At the beginning of each step a k -bit word is transmitted (stored) representing the value of each sample from the step; next we transmit the l bits representing the length of the step.

Example 3.11

In what follows we will present a method used in compressing facsimile black and white images; this method is extremely simple and quite efficient, although inferior to Huffman coding. A facsimile system with 1728 samples/line generates identical streams having $r_1 = 100$ or higher (generally for white streams), therefore the steps are encoded with $l = 8$ bits (Fig. 3.14).

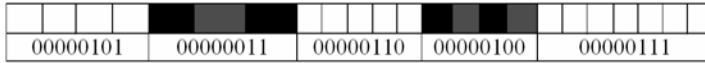


Fig. 3.14 Uniform steps coding for black and white images

Streams coding for texts

When we are interested in processing texts, a stream of repeating characters can be transmitted in a compressed form following the algorithm: a special symbol S is transmitted at the beginning of the repeating characters stream to indicate repetition, next the repeating character X and finally a character C which is counting the repetitions (Fig. 3.15).

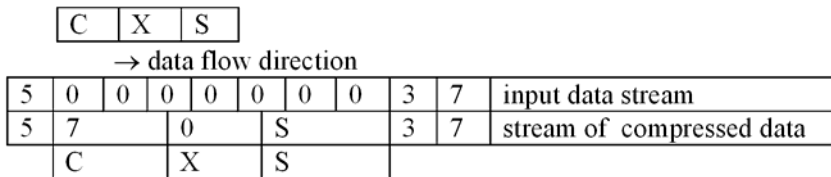


Fig. 3.15 Text compression

The stream of seven zeros was compressed transmitting CXS.

3.7.4 Comma Coding

Comma code is a uniquely decodable code which indicates through a symbol (0 or 1) the ending of every codeword.

Example 3.12

The source $S: \begin{pmatrix} s_1 & s_2 & s_3 & s_4 \\ 1/2 & 1/4 & 1/7 & 1/8 \end{pmatrix}$, is encoded using a comma code as:

S_i	p_i	c_{iC}	c_{iH}
S_1	1/2	0	0
S_2	1/4	10	10
S_3	1/8	110	110
S_4	1/8	1110	111

We get the average length and the compression ratio:

$$\bar{l}_C = \sum_{i=1}^m p_i l_i = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{8} \cdot 3 + \frac{1}{8} \cdot 4 = 1.87$$

$$R_{cC} = \frac{l_u}{l_C} = \frac{2}{1.87} \cong 1.07$$

The corresponding Huffman code is presented in the last column of the coding table (C_{iH}). In this case we have:

$$\bar{l}_H = 1.75$$

$$R_{cH} = \frac{2}{1.75} \cong 1.14 R_{cH}$$

It must be noticed that for the source under discussion the compression ratio is good enough compared to the maximum value obtained by Huffman coding.

Remark

For certain applications, comma coding can be more advantageous than Huffman coding due to the implementation simplicity and its robustness to errors (comma code is much more robust to errors unlike Huffman code). These advantages are paid by reducing coding efficiency (the compression ratio, respectively) in comparison to optimal codes, although in many cases the decrease can be small.

3.7.5 Dictionary Techniques [41]

Text files are characterised by the frequent repetition of some sub-streams. Numerous compression algorithms perform detection and removal of the repeating sub-stream.

The dictionary techniques build the dictionary of common sub-streams either at the same time with their detection or as a distinct step. Each dictionary sub-stream is put into correspondence with a codeword and the message is then transmitted through encoded sub-streams from the dictionary.

According to the input/output messages dimension, the dictionary techniques can be classified in:

- *fixed – fixed*: dictionary sub-streams have the same length and are uniformly encoded
- *fixed – variable*: sub-streams are uniform but are not uniformly encoded, for example using a Huffman code
- *variable – fixed*: at the compressor input the sub-streams are not uniform but are uniformly encoded
- *variable – variable*: lengths of the dictionary sub-streams and the coding is not uniform

After the dictionary has been built and the message encoded, the dictionary and the encoded message must be transmitted.

The compression efficiency is better when the dictionary is also compressed. This is possible if the compressor and the de-compressor simultaneously build up their dictionaries, in the ongoing compression process. This idea can be found in LZ-type algorithms.

3.7.6 Lempel-Ziv Type Algorithms (LZ)

Adaptive dictionary techniques

Dictionary techniques must allow the transmission of the dictionary from the compressor (transmitter), to the de-compressor (receiver). If the dictionary is static, it is not transmitted (static compression techniques). When using semi-adaptive dictionary algorithms, the first step is to transmit the dictionary and then to compress the message.

Adaptive dictionary techniques do not require explicit transmission of the dictionary. Both transmitter and the receiver simultaneously build their dictionary as the message is transmitted. At each moment of the encoding process the current dictionary is used for transmitting the next part of the message.

From adaptive dictionary techniques category, the most used ones are the algorithms presented by J. Ziv and A. Lempel in two papers published in 1977 and 1978; these algorithms are known as “LZ algorithms”.

The *basic idea* of LZ algorithms is to replace some sub-streams of the message with codewords so that for their each new occurrence only the associated codeword will be transmitted.

“Lempel and Ziv have hidden this brilliant idea in a sea of math” [41]. The works of these two mathematicians were extremely theoretical, becoming accessible due to other authors descriptions, so that LZ coding is in fact a family of algorithms built upon the same idea.

LZ-77 algorithm

LZ-77 algorithm uses the following two parameters:

- $N \in \mathbb{Z}[1, \infty]$, a window buffer length, which will move upon the transmitted message.
- $F \in \mathbb{Z}[1, N-1]$, the maximum length of encoded stream with $F \ll N$

Typical values used in practice are $N \cong 2^{13}$ and $F \cong 2^4$, both expressed as powers of two and making the algorithm easier for implementation.

The algorithm involves a window shifting register (buffer) of length N through which the message is shifted from right to left. The first $N-F$ elements form the

Lempel block, containing the most recently $N-F$ transmitted letters, and the next F elements form Ziv block containing the next F letters to be transmitted (Fig. 3.16).

	Lempel							Ziv					
← Message output	a	b	c	a	b	A	b	c	a	b	a	a	← Message input
	1	2					N-F	N-F+1			N		

Fig. 3.16 The LZ-77 algorithm illustration

At the beginning, the algorithm initialises Lempel block with a predefined stream and loads Ziv block with message's first characters. Encoding means to find the longest sub-stream from the register which has the first element in Lempel block and is identical to the stream starting from position $N-F+1$. This finding means to transmit a triplet (S, L, A) , where $S \in [1, N-F]$ is the position in the Lempel block from where the stream begins, $L \in [1, F]$ represents the length of the stream just found and A represents the letter where the similarity stopped. Next, the message is shifted from right to left in the shift register until the next letter, which has to be encoded, becomes the farthest element at the left from Ziv.

Some *remarks* regarding the algorithm ingenuity:

- the third element from the transmitted triplet A makes possible to avoid the situation when a letter does not exist in the Lempel block ($L = 0$)
- the stream from Lempel found to be similar to one from Ziv must start in Lempel block but could be able to extended into Ziv block
- the “dictionary” used in this technique is made up of every sub-stream from Lempel block; it must not be transmitted because it is dynamically resized both in encoder and decoder blocks
- the algorithm is locally adaptive because it uses $N-F$ previously transmitted letters, at most
- the search inside the table is linear and as a result, the encoding time is also dependent on the lengths N and F
- the decoding process is fast, because it doesn't involve a linear search inside the buffer
- due to the fact that N and F are finite numbers, S, L, A can be transmitted using an exact number of bits.

Example 3.13

An LZ77 coding example, starting from an empty Lempel buffer; the message is: “aabaabacabadad”

Lempel Buffer								Ziv				Transmitted code (S,L,A)		
1	2	3	4	5	6	7	8	9	10	11	12	S	L	A
0	0	0	0	0	0	0	0	a	a	b	a	0	0	a
0	0	0	0	0	0	0	a	a	b	a	a	8	1	b
0	0	0	0	0	a	a	b	a	a	b	a	6	4	0
0	a	a	b	a	a	b	a	c	a	b	a	0	0	c
a	a	b	a	a	b	a	c	a	b	a	d	2	3	d
a	b	a	c	a	b	a	d	a	d	a	d	7	4	0

Fig. 3.17 Illustration of LZ-77 algorithm

LZ-78 algorithm

LZ-78 version is similar to LZ-77, the difference being that the Lempel block is a continuously growing ‘dictionary’; having the dimension $d \in Z$ theoretically unlimited and the strings are numbered from 0 to d-1. Furthermore, there is no limit imposed to the Ziv block length.

For better understanding of the algorithm, let us consider Fig. 3.18:

0			
1	a		
2	b		Unlimited Ziv
3	ab	← Message output	aababc...
4	c		
5	bc	d = 6	

Fig. 3.18 Illustration of LZ-78 algorithm

At the beginning, the algorithm initialises the dictionary with an all zero stream and sets $d=1$. At each step, the algorithm sends the longest stream from Lempel which is identical to the one from Ziv (actually its associated code) and adds the next letter from Ziv where the similarity has been lost. This way, p=‘ma’ is transmitted by the equivalent of ‘m’ from the dictionary using $\lceil \log_2 d \rceil$ bits and by ‘a’ which is transmitted as it is. Next, the new stream (ma) is added to the dictionary, the message shifts in the Ziv block and the process starts all over again.

The decoder must decode $k+1$ streams to be able to add the k -th stream to the dictionary.

LZW algorithm (Lempel – Ziv – Welch ‘84)

In 1984, T. Welch published a version of LZ-78 algorithm. This version abandons the idea of transmitting the letter at which the similarity between the message

sub-streams and those from the ‘dictionary’ breaks. Explicit transmission of each stream last letter can be avoided by initialising the dictionary with all the symbols used in the message (e.g. the ASCII characters). This way, the letter that had to be transmitted explicitly will be the next stream first letter. The most important change of LZ-78 version is removing the explicit letter transmission. An improved version of the algorithm forms the base of the Compress Program under UNIX.

LZW algorithm is based on the *idea* of transforming stream symbols $S=s_1s_2s_3s_4\dots$ in sub-streams succession $S=S_1S_2S_3\dots$. Each sub-stream S_i (sub-stream containing more s_i symbols) will be associated to a fixed length codeword. Initially, we will have a coding table where a codeword is associated to each symbol of the source alphabet: $s_i \rightarrow c_i$. Next, the message will be read and when groups of symbols occur for the first time, they are added to the table (additional table) so that at a new occurrence they will be replaced with the codeword associated to their first occurrence.

LZW algorithm has the following steps:

- Encoding:
 - Initialise streams table with singular symbols (source alphabet) $S = \{s_i\}, i = \overline{1, M}$, and encodes them with fixed length words: $s_i \rightarrow c_i$.
 - Read the first symbol s_1 and sets $P=s_1$, where P is the prefix (found in the prefixes stream)
 - Read the next symbol: $E=s_k$
 - if PE can be found in the prefixes table, $P=PE$ is made
 - if PE is not in the prefixes table, PE is added to the streams table (supplementary table), and $E=P$ is added to the prefixes stream P
- Decoding:
 - the decoder will build the same streams table as the one from encoding
 - each word received is shifted, using the table from encoding, in a prefixes stream and the extension symbol PE (which is stored), and this is recursively repeated until the prefixes stream contains only one symbol

Example 3.14

Consider the source alphabet $S=\{a,b,c,d\}$, so that the streams table is initialised with:

s_i	c_i
a	1
b	2
c	3
d	4

Encoder is fed with stream $S = \text{"ababcbababaadaa"}$.

Encoding begins by reading the first symbol $s_1=a$ and setting $P=a$.

Next, the symbol s_2 is read, $E=b$ but because $PE=ab$ is not in the table of streams, the stream $PE=ab$ is added to the supplementary streams table, and $P=E=b$ is added to the prefixes stream P . The third symbol $s_3=E=a$ is read, but $PE=ba$ not being in the streams table, the stream PE is added to the supplementary streams table and $P=E=a$ is added to the prefixes stream P . For the fourth symbol $s_4=b=E$, we have $PE=ab$ in the additional streams table, therefore $P=PE$ and ab is added to the prefixes stream P a. s. o.

Decoding: the decoder will use the same streams table as the one used for compression. Each received codeword is moved from the similar prefixes table and an extension symbol (which is extracted and stored) is recursively repeated until the prefixes stream contains only one symbol.

Let us consider, for example, the case when we receive “9”:

$9 = 6b \rightarrow$ stores b
 $6 = 2a \rightarrow$ stores $a \quad \Rightarrow$ the decoded sequence is ab
 $2 = b \rightarrow$ stores b

Table 3.13 shows how the LZW algorithm works for the sequence given in Example 3.14.

Table 3.13 LZW algorithm applied to Example 3.14

Input message	a	b	a	b	c	b	a	b	a	b	a	a	d	a	a
Stream added (to the coding table – PE)		ab	ba		abc	cb		bab			baba	aa	ad	da	
Code added (to the coding table)		5	6		7	8		9			10	11	12	13	
Transmitted stream P		a	b		ab	c		ba			bab	a	a	d	aa
Transmitted code		1	2		5	3		6			9	1	1	4	11

Conclusions regarding the LZW algorithm

- LZW is an universal compression algorithm which can be applied to any source, without even knowing it in advance (it is an adaptive algorithm)
- coding/decoding are interleaved with a learning process developed while encoder/decoder builds and dynamically modifies the streams table; the decoder does not need coding tables, because it builds an identical table while receiving the compressed data.
- dividing the input message S into sub-streams S_i is not optimal (it doesn't take into consideration the source statistics), due to which LZW is not optimal; the idea of Shannon first theorem is used here, the coding being performed on symbols groups (on sub-streams).
- improved versions of LZW algorithm can be obtained making the followings:
 - setting a limit to the length of the streams included in the coding table
 - using a variable number of bits for codeword length (l), smaller at the beginning, then higher as the transmitted codewords number is increased, which implies the use of a signalling character.

- when coding table is full, there are two options: either to erase the table and start the process all over again or to continue with the same table but stopping the process of adding new characters.
- like other compression algorithms, LZW is sensitive to transmission errors; the most used method to ensure protection when saving compressed data on disks or magnetic tape is to apply the CRC (Cyclic Redundancy Check).

3.7.7 Arithmetic Coding

Before getting into details of arithmetic compression, let us present a new point of view upon the source coding process. As we have already shown, the information given by a symbol s_i is $-\log_2 p_i$ where p_i is symbol probability. Now let us imagine that this information is a cube of volume $-\log_2 p_i$. In this context, the codeword c_i associated to the symbol s_i will be a box through which that volume will be sent (transmitted through the channel). The problem is that we have only boxes of certain capacities and we want to work as efficiently possible (to avoid using big boxes for small volumes). In comparison with the uniform and entropic coding, in the first case we have boxes of fixed capacity (obviously a big waste of space), and in the second case we have boxes of different sizes. Looking at compression from this point of view, we get the idea of using boxes of the same capacity but filled with more cubes (the information of a symbol), in the order of their arrival.

From an informational point of view, the basic idea is to encode more symbols using only one codeword, of fixed length. However, the problem that must be dealt with is how to extract the symbols from the received codeword.

One solution to this problem is given by the arithmetic coding.

The steps of *arithmetic coding algorithm* are:

1. Put the symbols in decreasing order of probabilities: $p_1 \geq p_2 \geq \dots \geq p_n$.
2. Divide the interval $[0; 1)$ into n intervals of dimensions p_1, p_2, \dots, p_n .
3. Read the first symbol of the message and memorise its associated interval.
4. Divide this interval into n intervals, each of them proportional to p_1, p_2, \dots, p_n .
5. Read the next symbol and memorise the interval associated to it.
6. Continue the process following the same algorithm.
7. Transmit one number from the last memorised interval.

Example 3.15

Encode the message “abac” using an arithmetical code designed for the source S:

$$S: \begin{pmatrix} a & b & c \\ 1/2 & 1/3 & 1/6 \end{pmatrix}$$

A new problem that must be solved is how to encode the sub-unitary numbers that must be transmitted. One way to tackle this problem may be the following: choose a minimum of the memorised interval and when this interval becomes smaller than that minimum, transmit a number contained in the previous interval. The transmitted number will not be a sub-unitary one, but an integer representing a multiple of the minimum quantum, multiple which can be found in the memorised interval. In this

way, for example, if we consider the minimum interval $1/128$, for the previous encoding, we transmit 40 ($40/128$ is in the interval $[23/72, 1/3]$) in binary on 5 bits.

Encoding:

Processed Interval			Read symbol	Memorized Interval
$1/2$ (1-0)	$1/3$ (1-0)	$1/6$ (1-0)	a	$[0, 1/2)$
0 a	$1/2$ b	$5/6$ 1 c		
$1/2$ ($1/2 - 0$)= $1/4$	$1/3$ ($1/2 - 0$)= $1/6$	$1/12$	b	$[1/4, 5/12)$
0 a	$1/4$ b	$5/12$ $1/2$ c		
$1/2(5/12 - 1/4)$ = $1/12$	$1/18$	$1/36$	a	$[1/4, 5/3)$
$1/4$ a	$1/3$ b	$7/18$ $5/12$ c		
$1/2(1/3 - 1/4)$ = $1/24$	$1/36$	$1/72$	c	$[23/72, 1/3)$
$1/4$ a	$7/24$ b	$23/72$ $1/3$ c		

We transmit a number from the interval $[23/72; 1/3]$, for example $23/72$.

Decoding ($23/72$ is received):

Processed Interval			Interval containing $23/72$	Associated symbol
$1/2$ (1-0)	$1/3$ (1-0)	$1/6$ (1-0)		
0 a	$1/2$ b	$5/6$ 1 c	$[0, 1/2)$	a
$1/2$ ($1/2 - 0$)= $1/4$	$1/3$ ($1/2 - 0$)= $1/6$	$1/12$		
0 a	$1/4$ b	$5/12$ $1/2$ c	$[1/4, 5/12)$	b
$1/2(5/12 - 1/4)$ = $1/12$	$1/18$	$1/36$		
$1/4$ a	$1/3$ b	$7/18$ $5/12$ c	$[1/4, 1/3)$	a
$1/2(1/3 - 1/4)$ = $1/24$	$1/36$	$1/72$		
$1/4$ a	$7/24$ b	$23/72$ $1/3$ c	$[23/72, 1/3)$	c

3.8 Lossy Compression in Differential Coding

3.8.1 Differential Pulse Code Modulation (DPCM)

In PCM, a signal $x(t)$, voice, video signal or else is sampled at a rate slightly higher than the Nyquist rate: $f_s = 2f_M$. For example, in digital telephony with maximum frequency $f_M=3.4$ kHz the sampling frequency is $f_s = 8$ kHz. These samples are very correlated, the signal $x(t)$, not changing fast from a sample to the next one (it is a memory source). The corresponding PCM signal is therefore highly redundant. The removal of this redundancy before encoding is the *basic idea of differential PCM* (DPCM). This principle is illustrated in Fig. 3.19.

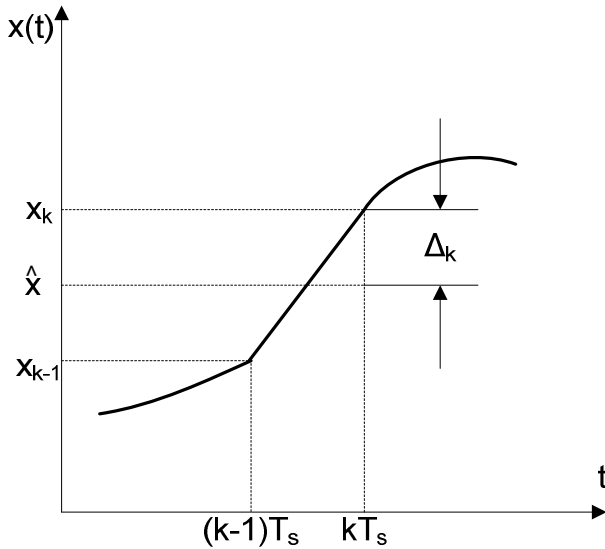


Fig. 3.19 Illustration of DPCM principle

In DPCM it is quantized the difference Δ_k between the sample x_k and the estimated value of it \hat{x}_k obtained from prediction from the previous samples of the signal $x(t)$. The prediction is possible only if $x(t)$ has memory, its statistical characteristics being known. For a pure random signal the prediction is impossible.

$$\Delta_k = x_k - \hat{x}_k \quad (3.71)$$

Further we will use the following notations:

x_k, \hat{x}_k, Δ_k for analogic sizes

$x_k', \hat{x}_k', \Delta_k'$ for digital sizes.

The pair DPCM modulator (encoder)-demodulator DPCM (decoder) is known as *DPCM codec*.

Remark

DPCM makes a lossy source compression, caused by the quantizing noise; this loss, for human users, if is under the thresholds of human audio sense (HAS) or human visual sense (HVS), is without quality loss.

Schemes for DPCM codecs are hundreds, thousands, depending on application and technology. Fig. 3.20 presents one block-scheme, illustrating the DPCM generation and detection.

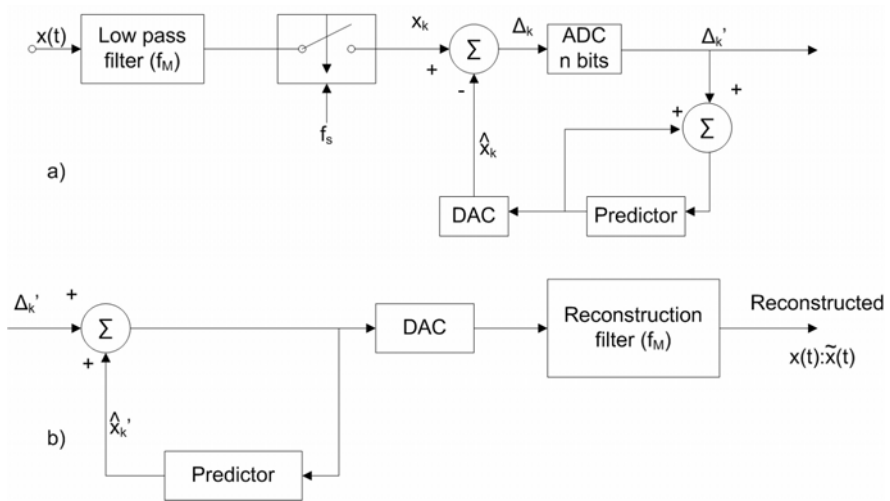


Fig. 3.20 DPCM codec: a) DPCM encoder; b) DPCM decoder

For a linear prediction we have:

$$\hat{x}_k = \sum_{i=0}^r a_i x_{k-1-i} \tag{3.72}$$

r being the prediction order and a_i coefficients chosen to minimize the prediction

mean square error: Δ_{kmin}^2 .

If $r=0$ (zero order prediction) we have:

$$\hat{x}_k = x_{k-1} \quad \text{and} \quad \Delta_k = x_k - x_{k-1} \tag{3.73}$$

Remark

DPCM is advantageous only if:

$$|\Delta_k|_{\max} < |X|$$

E.g.: suppose a zero order prediction and two values corresponding to two consecutive samples:

$$x_k = X \quad \text{or} \quad -X \quad \text{and} \quad x_{k-1} = -X \quad \text{or} \quad X;$$

it results that :

$$|\Delta_k| = 2|X|,$$

meaning that for the same quantization noise, the length of the corresponding digital word n_{DPCM} is higher than that one required in PCM (n_{PCM}).

The degree of compression is appreciated by the *compression ratio*:

$$R_c = \frac{n_{\text{PCM}}}{n_{\text{DPCM}}} = \frac{D_{\text{PCM}}}{D_{\text{DPCM}}} \quad (3.74)$$

with the mention that the sampling frequency is equal for both modulations:

Another quality parameter practically used is *prediction gain*:

$$G_p = \frac{P_x}{P_{\text{difference}}} = \frac{X_{\text{rms}}^2}{\Delta_{\text{krms}}^2} \quad (3.75)$$

A better compression is realized if adaptive DPCM is used (ADPCM). A fully adaptive DPCM system will adapt both the predictor and quantizer from the ADC.

DPCM is widely used for voice and video compression. For example, in digital telephony, a standard channel has $D_{\text{PCM}} = 64 \text{ kbps}$. Using a linear prediction coding (LPC [3]) the bit rate is diminished to 2.4 kbps, meaning a compression ratio of:

$$R_c = \frac{D_{\text{PCM}}}{D_{\text{DPCM}}} = \frac{64 \text{ kbps}}{2.4 \text{ kbps}} \approx 27.$$

3.8.2 Delta Modulation (DM)

Delta modulation is 1 bit DPCM of a zero order prediction, $r = 0$:

$$\Delta_k = x_k - x_{k-1} = \Delta \quad (3.76)$$

Being quantized with $q = 2$ number of levels ($n=1$):

$$\Delta_k' = \begin{cases} 1, & \text{if } x_k > x_{k-1} (+\Delta) \\ 0, & \text{if } x_k < x_{k-1} (-\Delta) \end{cases} \quad (3.77)$$

Consequently, the corresponding bit rate is:

$$D_{\text{DM}} = f_s \quad (3.78)$$

Remark

The sampling frequency (f_s) used in DM is much higher than the one used in PCM and DPCM in order to avoid an acceptable quantized SNR. The greatest advantage of the DM is its simplicity.

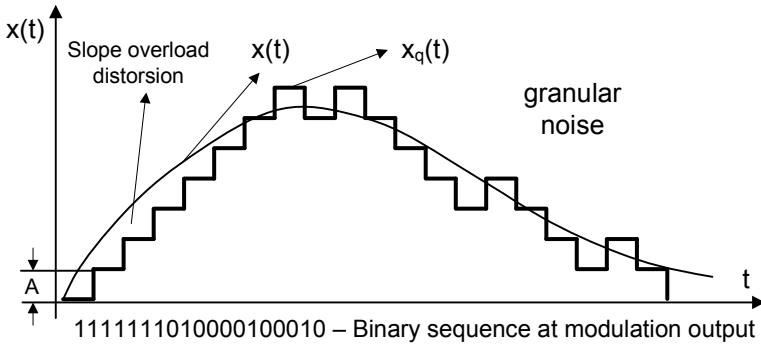


Fig. 3.21 Illustration of the DM

We have the following notations:

- $x(t)$ is the input signal
- $x_q(t)$ is the staircase approximation of the $x(t)$ signal

Noise in Delta Modulation

In any differential PCM modulation, occur two types of noise: the slope overload noise, meaning difficulties of following the signal (put into evidence by long series of “0” or “1”), and granular noise corresponding to constant segments of $x(t)$ and acting when 101010... occur.

Both types of noise will be illustrated for DM.

a) *Slope-overload noise (Fig. 3.22)*

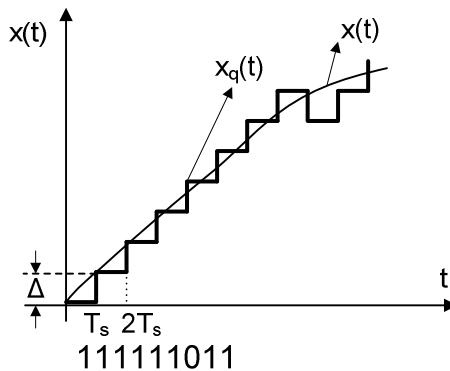


Fig. 3.22 Slope-overload noise

In order to eliminate this type of noise, the necessary condition is:

$$\max \left| \frac{dx}{dt} \right| \leq \frac{\Delta}{T_S} = \Delta \cdot f_S \tag{3.79}$$

The relation (3.79) indicates that the slope-overload condition requires large steps Δ .

Example 3.16

Assume a sinusoidal signal $x(t)$:

$$x(t) = X \sin 2\pi f t \tag{3.80}$$

The slope-overload condition in this case will be: $\frac{dx}{dt} = 2\pi f X \cos 2\pi f t$

$$\left| \frac{dx}{dt} \right|_{\max} = 2\pi f X < \Delta \cdot f_S \tag{3.81}$$

The relation shows the dependency of the slope-overload condition on frequency. This relation shows the limitations that differential modulations have at high frequencies.

b) Granular noise (Fig. 3.23)

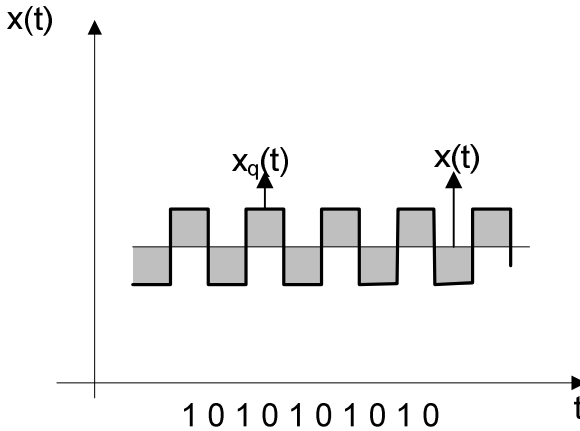


Fig. 3.23 Illustration of granular noise

Under the assumption of lack of the slope overload, the quantization noise n_q is a random signal with maximum amplitude $\pm\Delta$ and with uniform pdf.

$$f(n_q) = \frac{1}{2\Delta}$$

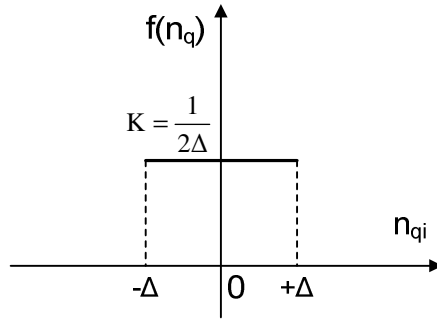


Fig. 3.24 Granular noise in DM - pdf representation

The power of the quantization noise is:

$$N_q = \frac{1}{R} \int_{-\Delta}^{+\Delta} n_q^2 f(n_q) dn_q = \frac{1}{R} \frac{\Delta^2}{3} \quad (3.82)$$

Remarks

- $N_q \approx \Delta^2$, meaning that to reduce the noise, a small Δ is required. This requirement to reduce the granular noise is opposite to the condition necessary for the slope overload condition (Δ large). A possible solution for both situations is the use of adaptive differential modulations [Jayant, Song..], in which Δ is adapted to the signal $x(t)$, thus, the SNR_q remains independent of $x(t)$.
- Comparing the quantized noise power in DM and PCM, it is obvious that:

$$N_{qDM} = \frac{1}{R} \frac{\Delta^2}{3} > N_{qPCM} = \frac{1}{R} \frac{\Delta}{12}$$

but it is important to remember that in DM, $f_S \gg 2f_M$ meaning that at the receiver, after the final low-pass filtering at f_M , only a part of N_{qDM} is delivered to the user (N'_{qDM}) [12]:

$$N'_{qDM} \cong \frac{f_M}{f_S} N_{qDM} = \frac{f_M}{f_S} \frac{\Delta^2}{3} \quad (3.83)$$

Consequently, at the user, the quantized SNR is:

$$SNR_{qDM} = \frac{P_X}{N_{qDM}} = \frac{X_{rms}^2}{\frac{f_M}{f_S} \frac{\Delta^2}{3}} = 3 \frac{f_S}{f_M} \frac{X_{rms}^2}{\Delta^2} \quad (3.84)$$

According to the relation (3.84), the following remarks could be done:

$$\begin{aligned} \text{SNR}_{\text{qDM}} &\sim X_{\text{rms}}^2 \\ \text{SNR}_{\text{qDM}} &\sim \frac{1}{\Delta^2} \\ \text{SNR}_{\text{qDM}} &\sim \frac{f_S}{f_M} \end{aligned}$$

Example 3.17

Assuming a sinusoidal signal (3.80), the slope-overload condition (3.81), fulfilled at limit, will give:

$$\Delta = \frac{2\pi Xf}{f_S} \tag{3.85}$$

Consequently, the quantizing SNR (3.84) will be:

$$\text{SNR}_{\text{qDM}} = \frac{3}{8\pi^2} \frac{f_S^2}{f_M} \frac{1}{f^2} \tag{3.86}$$

The relation (3.86) indicates the reverse proportionality of the quantized SNR with frequency, showing that the differential modulations are disadvantageous at high frequencies: $f \uparrow \Rightarrow \text{SNR}_{\text{qDM}} \downarrow$.

3.8.3 Delta-Sigma Modulation

One solution to eliminate the already mentioned disadvantage of differential modulations is to use delta-sigma modulation, which allow a quantized SNR independent of the signal frequency.

Before delta modulation, the signal $x(t)$ is integrated, and after delta modulation, by derivation, the initial spectrum of $x(t)$ is recovered.

The principle of delta-sigma modulation is illustrated in Fig. 3.25.

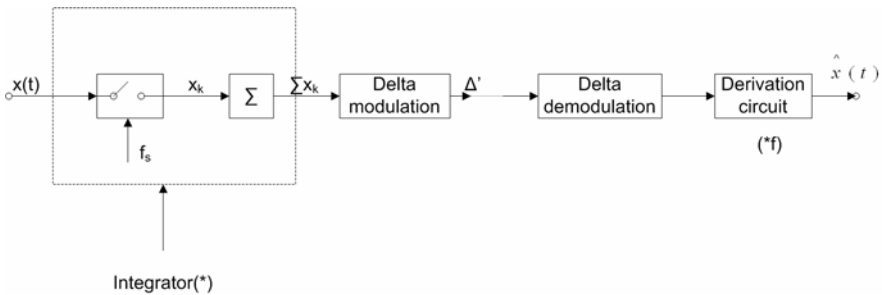


Fig. 3.25 Delta-Sigma modulation - demodulation block scheme

3.8.4 Comparison and Applications of Digital Modulations

Comparison of digital modulation is made always for the same signal $x(t)$. It is a non-sense to compare a voice codec with a television one. Comparison can be done in two ways:

- same bit rate ($\dot{D} = ct.$): the best modulation will ensure the greatest quantized SNR (the highest quality)
- same quality ($SNR_q = ct.$): the best modulation is that one which has the

smallest bit rate (\dot{D}) – best compression.

According to these criteria, the following remarks can be made:

- PCM is the most general digital modulation
- DPCM is the best for both criteria: highest SNR_q at minimum bit rate \dot{D} .
- DM is the simplest and obviously the cheapest, useful for applications in which the quality is not essential, but requiring robustness as military communications.

3.9 Conclusions on Compression

As showed in 3.2, the main purpose of source coding is the compression. This is accomplished by reducing the source’s decision rate as a consequence of decreasing the redundancy.

One classification of compression can be made as shown in figure 3.26.

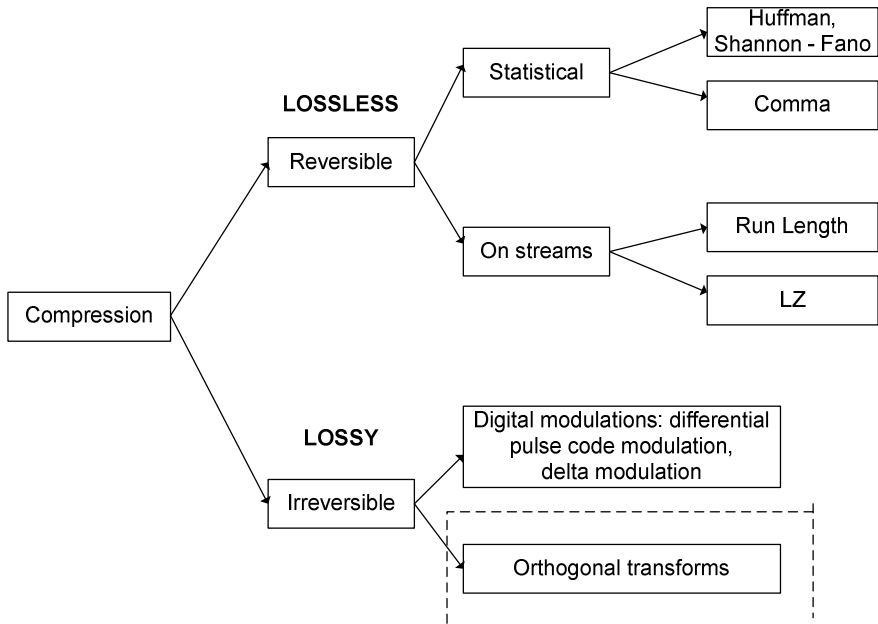


Fig. 3.26 Classification of compression

Reversible compression (lossless) is the compression that uses transforms that preserve the source entropy but decreases the redundancy.

Irreversible compression (lossy) is the compression that uses entropy-reducing transforms. To this category belong digital modulations (differential pulse-code modulation, delta modulation) and orthogonal transforms; in the present book were described only the digital modulations.

Statistic compression is the compression that requires the knowledge of source's statistics, $P(s)$; in this case the algorithms are also known as *static algorithms*. If $P(s)$ is determined as the first step and effective coding is performed after, the algorithm is semi-adaptive (quasi-dynamic). Thus, for unknown sources the use of statistic algorithms like Huffman, Shannon-Fano, implies, as a first step, to determine $P(s)$.

For memory sources, the lossless compression algorithms used are Run Length Coding and LZ. In these cases, streams are generated using groups of symbols, which act as a memoryless source and can be optimally encoded (e.g. run length coding with Huffman coding of the streams).

Dynamic algorithms (adaptive) such as dynamic Huffman or LZ are used for compressing unknown sources.

For memory sources, Markov models can also be used.

Some real applications of compression

Until quite recently the practitioner point of view [41] was to ignore data compression due to the following reasons:

- most compression techniques are unable to compress different data types.
- it is impossible to predict the compression degree.
- some of them “narrow-minded” and “afraid” of the “mysticism” that surrounds the maths incorporated in compression.
- “fear” of the complexity introduced by the level of compression.

All these have led to a big gap between research and practice which can be filled either by commercial pressure (the case of fax compression) or by a research effort to provide as simple as possible, more concrete aspects of the data compression techniques.

Current trends indicate a rapid expansion of the compression techniques, taking into consideration their applications: multimedia, HDTV, disk drivers, compression software, CD recordings etc. A comprehensive description of the domain is done in [31].

However, it must not be forgotten that besides its great advantages (decreasing the bit rate in transmission and reducing the space in storage), compression is extremely vulnerable to errors; this is the reason why, applications including compression require error protection.

References

- [1] Aaron, M.R.: PCM transmission in the exchange plant. *Bell System Technical Journal* 41(99), 99–141 (1962)
- [2] Apiki, S.: Lossless data compression. *Byte*, 309–314, 386–387 (1991)
- [3] Bellamy, J.: *Digital Telephony*, 2nd edn. John Wiley and Sons, Chichester (1991)
- [4] Berger, T.: *Rate Distortion Theory: Mathematical Basis for Data Compression*. Prentice-Hall, Englewood Cliffs (1971)
- [5] Berrou, C., Glavieux, A.: Near Shannon limit error-correcting coding and decoding turbo-codes. In: *Proc. ICC 1993, Geneva*, pp. 1064–1070 (1993)
- [6] Borda, M., Zahan, S.: A new algorithm associated to Huffman source coding. *Acta Technica Napocensis* 1, 61 (1994)
- [7] Bruce, A., et al.: *Molecular Biology of the Cell*, 5th edn. Garland Science (2008)
- [8] CCITT yellow book, *Digital Networks Transmission systems and multiplexing equipment*, vol. 3, Geneva, ITU (1981)
- [9] Chevalier, J., Gheerbrant, A.: *Dictionnaire des symboles*, Laffont, Paris (1995)
- [10] Claverie, J.-M., Notredame, C.: *Bioinformatics For Dummies*, 2nd edn (2007)
- [11] Crosier, A.: Introduction to pseudoternary transmission codes. *IBM Journal of Research and Development*, 354–367 (July 1970)
- [12] Fontoillet, P.G.: *Systemes de telecommunications*, Lausanne (1983)
- [13] Gallager, R.G.: *Information Theory and Reliable Communication*. John Wiley & Sons, Chichester (1968)
- [14] Gallager, R.G.: Variation on a theme by Huffman. *IEEE Transactions on Information Theory* IT 24, 668–674 (1978)
- [15] Graef, G.L.: Graphics formats. *Byte*, 305–310 (September 1989)
- [16] Hamming, R.: *Coding and Information Theory*. Prentice-Hall, Englewood Cliffs (1980)
- [17] Held, G., Marshall, T.R.: *Data compression-techniques and applications, Hardware and software considerations*. John Wiley and Sons, Chichester (1987)
- [18] Hedit, M., Guida, A.: Delay Modulation. *Proceedings IEEE* 57(7), 1314–1316 (1969)
- [19] Ionescu, D.: *Codificare si coduri*. Editura Tehnica, Bucuresti (1981)
- [20] Jayant, N.S., Noll, P.: *Digital Coding of Waveforms: Principles and Applications to Speech and Video*. Prentice-Hall, Englewood Cliffs (1984)
- [21] Johnston, B., Johnston, W.: LD-4. A Digital Pioneer in Action. *Telesis* 5(3), 66–72 (1977)
- [22] Johannes, V.I., Kaim, A.G., Walzman, T.: Bipolar pulse transmission with zero extraction. *IEEE Transactions on Communications Techniques* 17(2), 303–310 (1969)
- [23] Kay, S.M.: Fundamentals of statistical Signal Processing. In: *Detection Theory*, vol. II. Prentice-Hall, Englewood Cliffs (1998)
- [24] Knuth, D.E.: Dynamic Huffman coding. *Journal of Algorithms* 6, 163–180 (1985)
- [25] Lu, W.W., Gough, M.P.: A fast-adaptive Huffman coding algorithm. *IEEE Transactions on Communications* 41(4), 535–543 (1993)
- [26] Mateescu, A., et al.: *Manualul inginerului electronist, Transmisiuni de date*. Editura Tehnica, Bucuresti (1983)
- [27] Moon, T.K., Stirling, W.C.: *Mathematical Methods and Algorithms for Signal Processing*. Prentice-Hall, Englewood Cliffs (2000)
- [28] Rabiner, L., Schafer, R.W.: *Digital Processing of Speech Signals*. Prentice-Hall, Englewood Cliffs (1978)

- [29] Rabiner, L., Schafer, R.W.: *Introduction to Digital Speech Processing (Foundations and Trends in Signal Processing)*. Now Publishers Inc. (2007)
- [30] Radu, M.: *Telefonie numerica*. Editura Militara, Bucuresti (1988)
- [31] Sayood, K.: *Introduction to Data Compression*, 2nd edn. Morgan Kaufmann, San Francisco (2000)
- [32] Schena, M.: *Microarray Analysis*. John Wiley and Sons, Inc., Chichester (2003)
- [33] Spataru, A.: *Fondements de la theorie de la transmission de l'information*. Presses Polytechniques Romandes, Lausanne (1987)
- [34] Storer, J.A.: *Data Compression: Methods and Theory*. Computer Science Press, Rockville (1988)
- [35] Takasaky, Y., et al.: Optical pulse formats for fibre optic digital communications. *IEEE Transactions on Communications* 24, 404–413 (1976)
- [36] Van Trees, H.: *Detection, Estimation and Modulation Theory, Part. I*. John Wiley and Sons, Inc., Chichester (1968)
- [37] Vucetic, B., Yuan, J.: *Turbo codes*. Kluwer Academic Publishers Group, Dordrecht (2001) (2nd Printing)
- [38] Wade, G.: *Signal Coding and Processing*. Cambridge University Press, Cambridge (1994)
- [39] Wade, G.: *Coding Techniques: An Introduction to Compression and Error Control*. Palgrave Macmillan, Basingstoke (2000)
- [40] Welch, T.: A Technique for high-performance data compression. *Computer*, 8–19 (June 1984)
- [41] Williams, R.N.: *Adaptive data compression*. Kluwer Academic Publishers, Dordrecht (1990)
- [42] Wang, Q., Parrish, A.R., Wang, L.: Expanding the genetic code for biological studies. *Chemistry & Biology* 16(3), 323–336 (2009)
- [43] Xiong, F.: *Digital Modulation Techniques*. Artech House, Boston (2000)
- [44] Xie, J., Schultz, P.G.: Adding amino-acids to the genetic repertoire. *Curr. Opin. Chem. Biol.* 9(6), 548–554 (2005)
- [45] Ziv, J., Lempel, A.: A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory* IT 23(3), 337–343 (1977)

Chapter 4

Cryptography Basics

Motto: *Get to know yourself.
(aphorism written on
the frontispiece of
Apollo Temple in
Delphi)*

4.1 Short History of Cryptography

Information (religious, military, economic, etc) was always power, this is why the desire to protect it in order to be accessible only to some elites, dates since ancient times. There were discovered ciphertexts older than 4000 years, coming from the *Ancient Egypt*. There are records that ciphered writings were used since the 5th century BC in *Ancient Greece*. Thus the Spartan military used a stick (*scytalae*) around which narrow ribbon lather, papyrus or parchment was rolled, spiral by spiral, on which the clear text was written along the axis. After finishing the writing, the ribbon was unrolled, the message becoming meaningless. Only the person who had an identical stick (playing the role of secret key) with the one used at writing, could have read it.

The Greek historian *Polybius* (2nd century BC) is the inventor of a ciphering square table size 5x5, table found at the origin of a large number of ciphers, used in modern times.

Steganography (from the Greek words *steganos*, which means covered and *graphein*, meaning to write), the technique of hiding secret messages, has its origin in the same time. Herodotus, the ancient Greek historian (5th century BC), tells in “The Histories” that the tyrant of Miletus shaved the head of a trusty slave and tattooed it with a secret message and, after hair regrown, sent him to Athens to instigate a rebellion against the Persians [42].

In *Ancient Rome* the secrecy of political and military information was ensured by encryption, the most famous being *Caesar cipher*, used since the Gallic Wars (58-51 BC).

All the *Sacred Books* of each religion contain parables and encrypted messages.

Less known, the *Arabian contribution* to the development of cryptology is remarkably important. David Kahn one of the greatest historians in the domain, underlines in his book [41], that cryptology is born in the Arabic world. The first three centuries of *Islamic civilization* (700- 1000 AC), besides a great political and military

extension, are a period of intensive translations in Arabic of the greatest works of Greek, Roman, Indian, Armenian, Hebrew and Syriac Antiquity. Some of them were written in already dead languages, meaning that they represented in fact ciphertexts. To read such texts is nothing else than cryptanalysis. *Al Kindi*, an Arab scientist from 9th century is considered the father of cryptology, his book on this subject being, at this moment, the oldest available [41]. That time, the scientific language was Arabic. The Arabs took over the Greek and Indian knowledge and introduced the decimal numeral system with the Arabic numbers (numerals). The terms *zero*, *cipher*, *algorithm*, *algebra* are due to Arabs. Cipher comes from the Arabian “šifi” which is the Arabic translation of number, “zero” from Sanskrit language. When the concept “zero” was introduced in Europe, using that time the Roman numerals, it produced a great ambiguity. For this reason, those speaking unclear were baptised as “cipher speakers”. This meaningless sense of a message is known even today as cipher.

During the *Renaissance*, besides the revival of the interest of the ancient civilisations, ancient cryptography was discovered. The expansion of diplomatic relations among different feudal states determined the progress of cryptography. To that period belong:

- *Leon Battista Alberti* (1404- 1472), the inventor of the polyalphabetic cipher (*Alberti cipher*), using a cipher disk, considered the most important advance in cryptography since Antiquity.
- *Johannes Trithemius* (1462- 1516), the author of the famous encrypted book *Steganographia*, in three volumes.
- *Giovani Batista Bellaso* (1505- ?), the inventor of the polyalphabetic substitution with mixed alphabets, known later as the *Vigénère cipher*.
- *Giambattista della Porta* (1535- 1615), the inventor of the digramic substitution cipher.

The telegraph and radio inventions in the 19th century, as well the two World Wars from the past century, have been strongly stimulating the development of cryptographic algorithms and techniques [41].

The continuous development and spreading of computer use in our life, the existence and fast growing of communications networks, the existence of powerful databases, the introduction and development of e-commerce and web mail accounts indicate an uncommonly growth of the volume and importance of transmitted or stored data and implicitly of their vulnerability. Security, in these systems, refers to:

- eliminate the possibility of deliberately or accidentally data destruction
- guarantee communication confidentiality in order to prevent data leakage to unauthorized persons
- authentication of data and entity in order to prevent unauthorized person from introducing or modifying data into the system
- in some particular cases such as electronic funds transfers, contracts negotiations, the existence of an electronic signature is important to avoid disputes between the transmitter and receiver about the sent message.

All these objectives show a great expansion of cryptography from diplomatic, military, political domains to the civil area, with strong economical and social features.

4.2 Terminology

Cryptography from Greek *kryptos* meaning hidden, secret and *graph* while means writing is the science of secret writings.

Plaintext/ cleartext (M) is the message requiring to be protected. In cryptography it is known as text, however its nature: voice, image, video, data.

Ciphertext (C) is the nonsense shape of the protected plaintext, inaccessible for adversaries.

Encryption/ enciphering is the transformation (E) of the plaintext M into ciphertext: $E(M)=C$.

Decryption/ deciphering is the reverse transformation (D) of encryption, i.e. the determination of the cleartext from the ciphertext: $D(C)=D(E(M))=M$.

Both encryption and decryption are controlled by one or more *cryptographic keys* (ki).

Cryptographic algorithm/ cipher is the mathematical function or functions used for encryption (E)/ decryption (D).

Cryptanalysis is the art of breaking ciphers, the process of obtaining the plaintext or the decryption key from the ciphertext only.

Cryptology is the science of both cryptography and cryptanalysis.

Cryptographer is the person dealing with cryptography.

Cryptanalyst is the person dealing with cryptanalysis.

Cryptologist is the person dealing with cryptology.

Attack is the cryptanalytic attempt.

Cryptosystem is the system where an encryption/ decryption process takes place.

Steganography is the technique of hiding secret messages into harmless messages (hosts), such that the very existence of the secret message is hidden (invisible).

4.3 Cryptosystems: Role and Classification

The block- scheme of a cryptosystem is presented in Fig. 4.1.

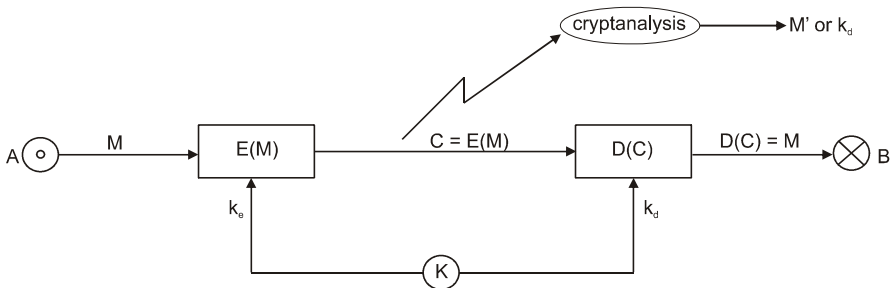


Fig. 4.1 Block-scheme of a cryptosystem where: A, B - entities which transmit, receive the information, E - encryption block, D - decryption block, M- plaintext, C - ciphertext, K - cryptographic key block, k_e - encryption key, k_d - decryption key

Role of a cryptosystem

The main tasks that a cryptosystem need to accomplish are:

- *Confidentiality/ secrecy/ privacy*: an unauthorized user should not be able to determine k_d from the ciphertext C even knowing the plaintext M . (Fig. 4.2); it means that the decryption (D) is protected.

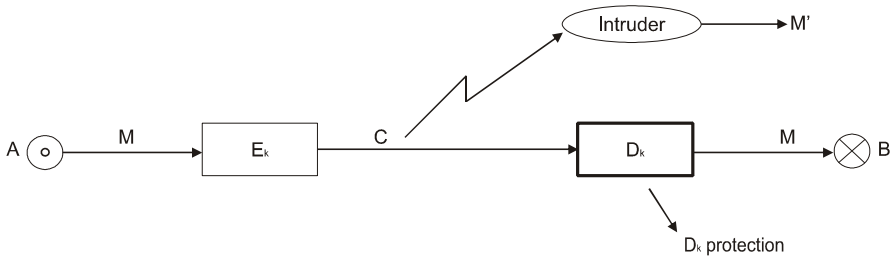


Fig. 4.2 Illustration of confidentiality

- *Authentication* (Fig. 4.3) is applied to:
 - entities (persons, terminals, credit cards, etc) and in this case is known as *entity authentication/ identification*.
 - information: the transmitted or stored information need to be authentic as origin, content, time of transmission or storage and defines *data. authentication/ data integrity*.

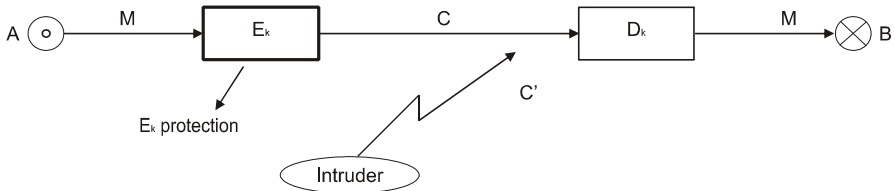


Fig. 4.3 Illustration of authentication

- *Digital signature* (S) is a mutual authentication (both data and entity authentication) User B must be certain that the received message comes precisely from A, being signed by A and A must be certain that nobody will be able to forge his signature.

Besides these three main objectives, the cryptosystem may ensure also, *nonrepudiation, authorization, cancellation, access control, certification, confirmation, anonymity*, etc. [64], [68].

Cryptosystems classification

According to the type of keys used for encryption (k_e), respectively for decryption (k_d), the cryptosystems are:

- *symmetric (conventional)*: use the same key for encryption and decryption.

$$k_e = k_d = k \quad (4.1)$$

and thus the key (k) is *secret* and requires a secure channel for transmission.

$$E_k(M) = C \quad (4.2)$$

$$D_k(C) = D_k(E_k(M)) = M \quad (4.3)$$

Remark

For simplicity the key is not always mentioned, thus is written:

$$E(M)=C \quad (4.2.a)$$

$$D(C)=D(E(M))=M \quad (4.2.b)$$

- *asymmetric (public-key)* use distinct keys for encryption (k_e) and decryption (k_d):

$$k_e \neq k_d \quad (4.4)$$

The encryption key, k_e , is public, whereas the decryption key, k_d , is private (secret). Encryption is done with the public key of the correspondent X (k_{eX}) and decryption is made using the private key of the correspondent X (k_{dX}). For simplicity the notation will be:

$$E_X = E_{k_{eX}} \quad (4.5)$$

$$D_X = D_{k_{dX}} \quad (4.6)$$

Thus, if A wants to communicate confidentially with B using a *public key cryptography (PKC)*, the protocol will be:

- 1) A will encrypt the message M with the public key of B

$$E_B(M) = C \quad (4.7)$$

- 2) B will decrypt the cryptogram C using its private key

$$D_B(C) = D_B(E_B(M)) = M \quad (4.8)$$

4.4 Cryptanalytic Attacks and Algorithms Security

Cryptanalytic attack is the attack made on the ciphertext in order to obtain the cleartext or the key necessary for decryption. Basically there are six types [64]:

1. *ciphertext only attack*: the cryptanalyst has the ciphertext $C_i = E_k(M_i)$ and must obtain the plaintext M_i or the decryption key.
2. *known plaintext attack*: the cryptanalyst has the ciphertexts C_i and the corresponding plaintexts M_i and needs to find the key k or the algorithm to obtain M_{i+1} from $C_{i+1} = E_k(M_{i+1})$.
3. *chosen plaintext attack*: it is possible to choose some plaintexts M_i and to know the corresponding cryptograms: $C_i = E_k(M_i)$; the cryptanalyst must find the key k or the algorithm of giving M_{i+1} from $C_{i+1} = E_k(M_{i+1})$.
4. *adaptive chosen plaintext attack*: the plaintexts M_i can be chosen and are adaptive to the previous cryptanalytic attacks results and the ciphertexts $C_i = E_k(M_i)$ are known. It is required the key k or the algorithm to obtain M_{i+1} from $C_{i+1} = E_k(M_{i+1})$.
5. *chosen ciphertext attack*: the cryptanalyst can choose ciphertexts C_i and plaintexts $M_i = D_k(C_i)$, his task being that of finding the decryption key k ; this attack is used mainly in PKC (Public Key Cryptography).
6. *rubber-hose cryptanalysis/ purchase key attack*, when the key is obtained without cryptanalysis (blackmail and robbery being the main ways); this is one of the most powerful attacks.

Algorithms security

In 19th century, the Dutch Auguste Kerckhoffs stated the fundamental concept of cryptanalysis, known as *Kerckhoffs law*: *the security of a cryptosystem need to lie only in the key*, the algorithm and the implementation being known. It was reformulated later [66] by Claude Shannon “as the enemy knows the system” and is opposite to the principle “security by obscurity”.

Different algorithms provide different degree of security, according to the difficulty of breaking them [64].

1. total break (zero security): the key is known
2. global deduction: a cryptanalyst finds an alternative algorithm equivalent to $D(C)$ without the knowledge of the key.
3. local deduction: the cryptanalyst finds the cleartext from the intercepted ciphertext.
4. information deduction: the cryptanalyst gets some information about the key or the cleartext.
5. computationally secure (strong) is the algorithm that cannot be broken with available resources in present and in the near future.
6. unconditionally secure is the algorithm impossible to be broken.

One time pad (OTP) is the only unconditionally secure algorithm. It was invented in 1917 by *Gilbert Vernam* and in 1949 Claude Shannon proved OTP ensures *perfect secrecy* [66]:

$$H(M)=H(M/C) \quad (4.9)$$

where $H(M)$ is the a priori entropy of the plaintext and $H(M/C)$ is the a posteriori entropy (conditional entropy of M).

OTP is a random key with at least the same length as the plaintext (M) and is used only once.

All other algorithms are breakable trying all the possible keys for a given ciphertext, until the plaintext is meaningful. This is the *brute force attack* (ex. If the key is n long, the number of possible keys is 2^n).

Complexity of an attack [64] can be expressed in different ways:

- a) *data complexity* represents the amount of data required to fulfil the attack.
- b) *processing complexity (work factor)* is the time necessary to perform the attack.
- c) *storage complexity* is given by the required memory to do the attack.

Rule: Complexity of an attack = $\min\{a, b, c\}$

Remark

Many attacks are suitable for parallelisation, meaning that the complexity can be substantially reduced (see the possibility to use the graphics processor units - GPU, for cryptanalytic purposes, based on its parallelism).

In cryptanalysis, when appreciating the size of an attack, we need to keep in mind *Moore law* [64]: *computing power doubles approximately at 18 months and the costs diminish 10 times every five years.*

4.5 Classic Cryptography

4.5.1 *Definition and Classification*

By *classic cryptography* usually is understood the cryptography since Antiquity until modern times, defined by Shannon work “Communication Theory of Secrecy Systems” published in 1949 in the Bell System Technical Journal.

Classic ciphers are symmetric ciphers, meaning the use of the same key, which is secret, for encryption and decryption. Basically they are hiding the plaintext using some elementary transformations.

- *substitutions*, in order to create confusion (a very complex relation between the plaintext and the ciphertext)
- *transpositions*, made by permutations, in order to diffuse the plaintext redundancy into the statistics of the ciphertext.

Some algorithms (ciphers) are applying these transformations iteratively in order to enhance the security.

According to the type of transformation, classic ciphers are classified as given in Table 4.1:

Table 4.1 Classification of classic ciphers

	Type of transformation	Variety	Cipher name															
C L A S S I C	Substitution	monoalphabetic	Caesar															
			Polybius															
		omophonic	Example: E $\left\{ \begin{array}{l} \rightarrow B \\ \rightarrow J \\ \rightarrow Q \end{array} \right.$															
		polygramic	Playfair															
		polyalphabetic	Trithemius Vigénère															
C I P H E R S	Transposition	<p style="text-align: center;">→ Ex:</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="border: none;">→</td> <td>V</td> <td>E</td> <td>N</td> <td>I</td> </tr> <tr> <td style="border: none;">↓</td> <td>V</td> <td>I</td> <td>D</td> <td>I</td> </tr> <tr> <td style="border: none;"></td> <td>V</td> <td>I</td> <td>C</td> <td>I</td> </tr> </table> <p style="text-align: center;">→VVVEIINDCIII</p>	→	V	E	N	I	↓	V	I	D	I		V	I	C	I	ADFGVX
→	V	E	N	I														
↓	V	I	D	I														
	V	I	C	I														
	Rotor machines	electro-mechanical devices implementing Vigénère versions	Enigma															

Substitution ciphers: a character or a group of characters of the plaintext is substitute for another character or group of characters in the ciphertext.

Transposition ciphers: the plaintext remains unchanged in the ciphertext, only the order of the characters is changed (a permutation is done).

Rotor machines is an electro-mechanical device, invented in 1920 (a continuation of Leon Battista Alberti idea used in his cipher disk) used to automatize the complicated iterated substitutions and permutations. Such a device contains a keyboard and a set of rotors, each one allowing the implementation of a Vigénère version. The most famous rotor machine is Enigma, used during World War II [41].

In what follows, the most famous classic ciphers will be presented.

4.5.2 Caesar Cipher

Caesar cipher is a monoalphabetic substitution cipher. Each letter of the plaintext is replaced by a new one obtained shifting three positions to right. The encryption is given by the relation:

$$C_i = E(M_i) = (M_i + 3) \bmod 26 \quad (4.10)$$

and decryption is given by:

$$M_i = D(C_i) = (C_i - 3) \bmod 26, \quad (4.11)$$

26 being the dimension of the Latin alphabet.

Caesar cipher is represented in Table 4.2.

Table 4.2 Caesar cipher

Clear text	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Cipher text	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Example 4.1

Using Table 4.2, the well known sentence becomes the ciphertext:

Cleartext:	V E N I	V I D I	V I C I
Ciphertext	Y H Q L	Y L G L	Y L F L

Using the same Table 4.2, the deciphering is at once: ciphertext \rightarrow plaintext.

Caesar cipher can be generalized:

$$C_i = (M_i + k) \bmod 26 \quad (4.12)$$

$$M_i = (C_i - k) \bmod 26 \quad (4.13)$$

where k , the shifting, can take 25 values:

$$k = \overline{1, 25} \quad (4.14)$$

meaning 25 possible keys.

Remarks

1. Caesar cipher is a simple linear congruential generator [64]:

$$X_n = (aX_{n-1} + b) \bmod N \quad (4.15)$$

A linear congruential generator is a pseudo-random generator given by relation (4.15), where a , b , N are constants and signify:

- a- multiplier (1 for Caesar cipher)
- b- increment (3 for Caser cipher)
- N- modulus (26 for Caesar cipher)

2. Caesar and generalized Caesar ciphers are not at all secure, the cryptanalysis being very quick. The brute force attack consists of trying all the 25 keys (generalized Caesar), until the message becomes meaningful. If the alphabet of the plaintext is not known, the cryptanalysis is not easy. If the encryption is applied after compression, cryptanalysis is much harder. For example if the plaintext is in the first step compressed using ZIP file format [68] which has an alphabet completely different (Latin letters, Greek letters, geometric symbols, etc.), and then encrypted using a simple substitution cipher, the brute force attack will not be successful.

3. monoalphabetic substitution ciphers, for which the alphabets of the plaintext and ciphertext are known, are very sensitive to *relative frequency analysis* [68]. For them, the relative frequency of letters in the ciphertext is the same with that of the plaintext (which is known if the language is known), meaning that the cryptanalysis is quite simple, and the steps are:
 - determination of relative frequency of letters in the a ciphertext.
 - comparison with the relative frequency of letters from the plaintext (known), resulting a sort of “skeleton” of the cleartext.
 - refinement of the analysis by search, until the result is a meaningful plaintext.
4. ways to avoid the relative frequency analysis, so to make more difficult the cryptanalysis, are: to use omophonic substitutions, polygramic or polyalphabetic in order to make as possible, an uniform relative frequency for letters in the ciphertext (maximum entropy).
5. a particular case of generalized Caesar cipher is *ROT13*, having $k=13$; thus ROT13 is its own inverse: applied twice, it gives the clear text.

$$C = \text{ROT13}(C) \quad (4.16)$$

$$M = \text{ROT13}(C) = \text{ROT13}(\text{ROT13}(M)) \quad (4.17)$$

Remark

This cipher is not used for security purposes (which is extremely poor), but for hiding potentially offensive messages (at users stations, in networks); it was in use in the early 1980, in the net.jokes newsgroup.

4.5.3 Polybius Cipher

The Ancient Greek historian Polybius (203-120 BC), being responsible with the operation of a “telegraph” used to send at distance messages, invented a substitution cipher, known since that as *Polybius square* (Table 4.3). The letters of the Latin alphabet (26) are arranged in a square of size 5x5. The letters I and J are combined in a unique character because the choice between them can be easily decided from the text meaning. The encryption consists of replacing each letter with the corresponding pair of numbers (the line and column crossing point).

Table 4.3 Polybius square

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Example 4.2

Using Polybius square (Table 4.3), the message S O S is encrypted 43 34 43.

Remarks

- this substitution cipher is using distinct alphabets for plaintext (letters) and ciphertext (numerals).
- the cipher can be changed, changing the disposal of letters in the square, which plays the role of key (see Playfair cipher).

4.5.4 Playfair Cipher

Playfair cipher is one of the most famous polygramic ciphers.

A *polygramic cipher* is using substitution of a group of characters in the plaintext alphabet, known as “poligram”, with other groups of characters, for example:

$$ABA \rightarrow RTQ,$$

in order to obtain uniform relative frequencies for all the letters in the ciphertext, and thus to heavier the cryptanalysis.

Playfair cipher was invented by Sir Charles Wheatstone in 1854, but is carrying the name of Lord Playfair, who was the promoter of its application [67].

- It uses a square of dimension 5x5 (as Polybius square), made using a keyword.
- The square is filled with the letters of the keyword (without repeating letters) downwards, from left to right; the letters I,J are put in the same box, as in Polybius square.

Encryption is done on groups of two letters, known as “*digram*”:

- if the letters of the diagram are in the same line (l), the right side character is taken.
- if the letters are in the same column (c) the bottom neighbour is taken.
- if they are on different lines and columns the character found at the crossing point of line (l) and column (c) is chosen.

Example 4.3

Using Playfair cipher, encrypt the plaintext CRYPTOGRAPHY, using the keyword LEONARD.

Solution

Table 4.4 Illustration of Playfair cipher

CR YP TO GR AP HY plaintext

· · · · ·
· · ↓ · · ·
· · · · ·

FD VT SN PG LU KW ciphertext

L	E	O	N	A
R	D	B	C	F
G	H	IJ	K	M
P	Q	S	T	U
V	W	X	Y	Z

Playfair cipher advantages

Playfair cipher has some advantages compared to the monoalphabetic substitution ciphers:

- the alphabet is the same ($N=26$ letters), but the number of digrams is: $26 \times 26 = 676$, meaning that the identification of distinct digrams is harder.
- the relative frequency analysis is made more difficult
- reasonably fast
- requires no special equipment.

For his advantages, long time Playfair cipher was thought unbreakable.

The first break of Playfair cipher was given in 1914 by Joseph O. Maubergne, officer in US Army, and coinventor with Gilbert Vernam of OTP.

Playfair cipher was extensively used by British Empire in the Second Boer War (1879- 1915) and World War I, and also by Australians and Germans in World War II [41].

4.5.5 Trithemius Cipher

Trithemius cipher belongs to polyalphabetic substitution ciphers.

A *polyalphabetic substitution cipher* is composed of more simple substitution ciphers. It was invented by *Leon Battista Alberti* (1404- 1472), one of the greatest Renaissance humanist polymaths (from the Greek *polymates* meaning “having learned much”): architect, poet, priest, linguist, philosopher and cryptographer. He is also the inventor of the *cipher disk*, named also *formula*. This device, the precursor of rotor machine, is the first polyalphabetic substitution example, using mixed alphabet and variable period. It consists of two concentric disks, fixed together by a pine and able to rotate one with respect to the other.

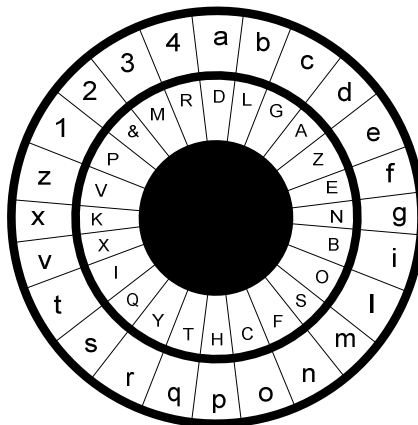


Fig. 4.4 Alberti cipher disk (formula)

Johannes Trithemius (1462- 1516), born in Trittenheim, Germany, was a Benedictine abbot, historian, cryptographer and occultist. As cryptographer he is

famous for the cipher carrying his name and for his work *Steganographia* in three volumes, written in 1499 and published in Frankfurt in 1606, an encrypted work, the last volume being recently decrypted, this is why a great aura of occultism surrounded his author. After being totally decrypted, it showed that the whole book deals with cryptography and steganography.

Trithemius cipher is a polyalphabetic substitution cipher. The 26 letter Latin alphabet is disposed in a rectangular square of dimension 25x25, known as *tabula recta* (Tab. 4.5). Each row, numbered from 0 to 25, contains the 26 letters alphabet cyclically shifted to the right. The row numbered with 0 is the alphabet in initial order (without shifting). The row numbered with 1 is a cyclic shifting of the previous row (0), with one position to the right, and so on. The encryption is as follows: the first character of the plaintext is encrypted selecting it from the first row, the second character from the second row and so on.

Table 4.5 Tabula recta of Trithemius cipher

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
0	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
2	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
3	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
4	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
5	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
6	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
7	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
8	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
9	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
10	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
11	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
12	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
13	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
14	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
15	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
16	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
17	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
18	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
19	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
20	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
21	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
22	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
23	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
24	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
25	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

Example 4.4

	1 2 3 4	5 6 7 8	9 10 11 12
Plaintext:	V E N I	V I D I	V I C I
Ciphertext:	W G O M	A O K Q	E S N U

4.5.6 Vigènère Cipher

Vigènère cipher is a polyalphabetic substitution cipher using Trithemius tabula recta and a keyword which command row selection for encryption and decryption.

This method belongs in fact to the Italian cryptologist *Giovan Battista Bellaso* (1505- ?) and in 19th century was assigned to *Blaise de Vigènère* (1523- 1596), a French diplomat and cryptographer. Vigènère published the original cipher of Battista Bellaso and proposed a much stronger key, such that this cipher was considered unbreakable (le *chiffre indéchiffrable*) till 19th century when the first break was published.

Being strong and simple enough, if used with cipher disks, it was used even during the American Civil War (1861- 1865).

Gilbert Vernam tried to repair the broken Vigènère cipher (creating the Vernam- Vigènère cipher (1918), unfortunately still vulnerable to cryptanalysis. But Vernan work led to the OTP, the unbreakable cipher, as demonstrated by Cl. Shannon [66].

Example 4.5

A Vigènère encryption of the plaintext VENI VIDI VICI using the key MONA is presented in Table 4.6.

Table 4.6 Vigènère encryption with key word

Keyword	M	O	N	A	M	O	N	A	M	O	N	A
Plaintext	V	E	N	I	V	I	D	I	V	I	C	I
Ciphertext	H	S	A	I	H	W	Q	I	H	W	P	I

An improved version of the presented Vigènère cipher is using a *trial key*. The trial key indicates the beginning line or lines for the first character or characters of the plaintext. Afterwards, the characters of the plaintext are used as keys. This means a feedback in the encryption process, the ciphertext being conditioned by the message itself.

Example 4.6

A Vigènère encryption with trial key: letter D and plaintext key word of the message:

VENI VIDI VICI,

is presented in Table 4.7

Table 4.7 Vigenère encryption with trial- key letter and plaintext keyword

Keyword	D	V	E	N	I	V	I	D	I	V	I	C
Plaintext	V	E	N	I	V	I	D	I	V	I	C	I
Ciphertext	Y	Z	R	V	D	D	L	L	D	D	K	K

Another version of Vigenère cipher is *Vigenère ciphertext keyword*. A trial-key is selected and afterward the characters of the ciphertext become keywords.

Example 4.7

A Vigenère encryption with trial-key letter D and ciphertext keyword of plaintext:

VENI VIDI VICI,

is given in Table 4.8.

Table 4.8 Vigenère encryption with trial-key and ciphertext keyword

Keyword	D	Y	C	P	X	S	A	D	L	G	O	Q
Plaintext	V	E	N	I	V	I	D	I	V	I	C	I
Ciphertext	Y	C	P	X	S	A	D	L	G	O	Q	Y

Remarks

- even though every character used as key can be found from the previous character of the ciphertext, it is functionally dependent on all the previous plaintext characters, including the trial key; the result is the *diffusion effect* of the statistical properties of the plaintext over the ciphertext, making thus the cryptanalysis very hard.
- for present security requirements, Vigenère cipher schemes are not very reliable; Vigenère important role lies in the discovery of non-repetition sequences generation (as key) using the message itself (plaintext or ciphertext) or parts of it.

4.6 Modern Symmetric (Conventional) Cryptography

4.6.1 Definitions and Classification

The modern cryptography is considered to start with Claude Shannon work “*Communication Theory of Secrecy Systems*”, published in 1949 in Bell System Technical Journal, based on a classified version of “*A Mathematical Theory of Cryptography*” presented in September 1945 for Bell Telephone Labs [65].

In these works he gave the mathematical model of a cryptosystem, the conditions for a perfect secrecy, the mathematical definition of the basic processing in encryption: confusion and diffusion, the mathematical description of a product cipher.

Assume that all the plaintexts M_i , with $i = \overline{1, m}$ and their a priori, probabilities p_i are known. By encryption, each M_i is transformed in a ciphertext C_i . Cryptanalysis tries to calculate from C_i the a posteriori probabilities of different plaintexts: $p(M_i/C_i)$.

A cryptosystem is *perfectly secret* if:

$$p(C_i/M_i) = p(C_i), \quad \forall i = \overline{1, m} \quad (4.18)$$

$$p(M_i/C_i) = p(M_i), \quad \forall i = \overline{1, m} \quad (4.19)$$

m being the number of distinct plaintexts (M_i), respectively ciphertexts (C_i). It means that the a posteriori probabilities equal the a priori ones.

These conditions imply the followings:

- 1) the number of keys (k_i) \geq the number of plaintexts (M_i) (4.20.a)

- 2) the length of keys (k_i) \geq the length of plaintext (M_i) (4.20.b)

- 3) a key need to be used only once (4.20.c)

These three conditions define the *one time pad (OTP) principle*.

The average quantity of information per plaintext M_i , respectively ciphertext C_i is the corresponding entropy:

$$H(M) = -\sum_{i=1}^m p(M_i) \log_2 p(M_i) \quad (4.21)$$

$$H(C) = -\sum_{i=1}^m p(C_i) \log_2 p(C_i) \quad (4.22)$$

As shown in chapter 2, the maximum value of the entropy (the decision quantity D , (2.14)) is obtained when the messages are equally probable:

$$H(M) \leq D(M) = \log_2 m \quad (4.23)$$

This information is completely hidden [3], [65] when the non-determination of the key is the greatest, which means that:

$$H(C) = \log_2 m \quad (4.24)$$

Relation (4.24) shows that the maximum non-determination introduced in a cryptosystem cannot be greater than the size of the key space: $\log_2 m$. It means that a cryptosystem using n bits length key will give $m = 2^n$ distinct keys (the *brute force attack dimension*). As $H(C)$ is greater, meaning m , respectively n greater, the breaking of a cryptosystem is harder.

Notice

Cryptanalysis is using redundancy of the plaintext (see 2.9.3). As known (see chapter 3), compression removes redundancy, this is why in real cryptographic implementation [3], [64], the first step is to compress the plaintext and only after that to encrypt it.

The basic processing in cryptography was defined mathematically also by Shannon:

- *confusion* defines the relation between the plaintext M_i , and ciphertext C_i which need to be as much possible complex and confused (chaotic); *substitution* (*S box* in modern ciphers) was identified as a primary confusion mechanism.
- *diffusion* [64] refers to the property of diffusing the redundancy of the plaintext M_i into the statistics of the ciphertext C_i ; *transposition*, made by *permutation* (*P box* in modern ciphers), and *linear transformations in finite fields* were identified as diffusion mechanisms.

Product cipher (SPN- Substitution- Permutation- Network) defines the alternation of S and P boxes during r *rounds* (iterations) were explained mathematically also by Shannon [3], [65].

Classification of modern symmetric ciphers, is made according to the type of information processing (as in error control coding - see chapter 5):

- *block ciphers*: the cleartext M is processed in blocks M_i of constant length n , each block M_i being encrypted once; the result is the ciphertext C_i of the same length n .
- *stream ciphers*: the encryption is character oriented ($n=1$ character), being in fact modern Vigénère versions.

4.6.2 Block Ciphers

4.6.2.1 Main Features

- the cleartext M is divided in blocks of constant length n (usually $32 \div 128$ bits), each block M_i being independently encrypted.
- each block cipher is a product cipher of S and P boxes, iterated r times; the number of iterations (rounds) r varies: 1 in classic ciphers, 16 in DES, Blowfish, 8 in IDEA, 10, 12 or 14 in AES, 48 in T-DES, etc. As r is greater, the cryptanalysis is harder, but also the speed of execution is greater. This is why a trade between security and speed of executing is done, the accepted value in most applications being $r_0 = 16$.

P box (Fig. 4.5) ensures the diffusion by permutation.

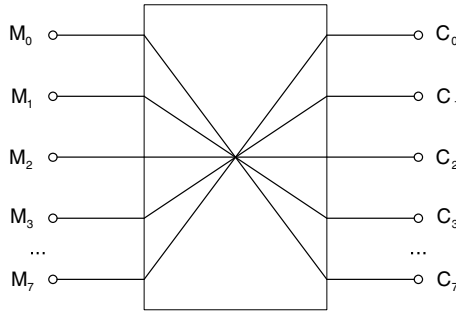
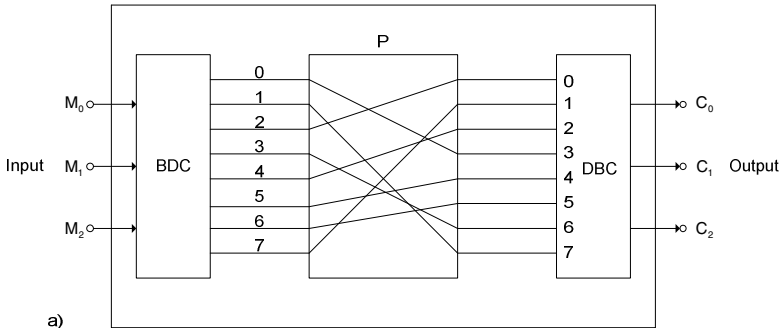


Fig. 4.5 Example of P box with $n=7$

The transformations made by a P box are linear; the internal links (permutations) being easily determined putting a binary “1” at each input and finding the corresponding output; if the input is of size n , the possible permutations of P box are $(2^n)!$

S box, which ensures the confusion, is usually composed of a binary to decimal converter (BDC) at the input, followed by a P box making the transposition between decimal positions and having a decimal to binary converter (DBC) at the output. An example is given in Fig. 4.6.



a)

Input	M_0	M_1	M_2	C_0	C_1	C_2	Output
0	0	0	0	0	1	0	3
1	1	0	0	1	1	1	7
2	0	1	0	0	0	0	0
3	1	1	0	0	1	1	6
4	0	0	1	0	1	0	2
5	1	0	1	0	0	1	4
6	0	1	1	1	0	1	5
7	1	1	1	1	0	0	1

b)

Fig. 4.6 Example of S box with $n = 3$: a) block scheme, b) truth table

The operation of such a box is as follows:

- the plaintext is expressed in binary in n bits and is applied to the input of binary to decimal converter (BDC)
- the P box, situated between the two converters, make a permutation of the 2^n inputs, the total number of combinations being $(2^n)!$

In our example (Fig. 4.6), n being 3, there are $(2^3)! = 40320$ possibilities. If n is great, for example 128, the cryptanalysis is, with conventional technology, impossible today.

A typical block cipher is a *product cipher*, made of an alternation of P and S boxes, known as SPN (*Substitution Permutation Network*) is given in Fig. 4.7:

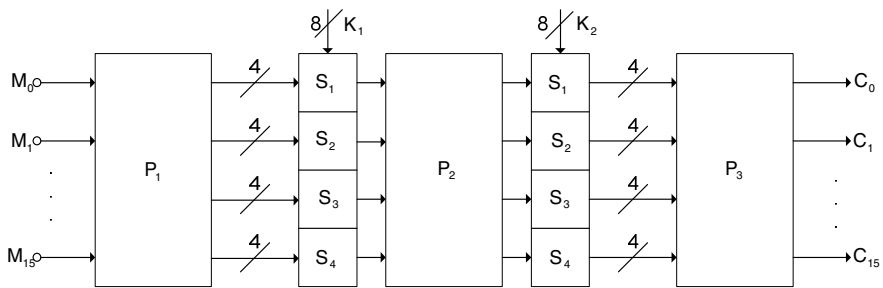


Fig. 4.7 Example of a product cipher (alternation of P and S boxes)

As illustrated in Fig. 4.7, P boxes are fixed (without key) and are used to implement the diffusion by permutation. S boxes receive the permuted text (for each box 4 bits) and also a key of 2 bits long to command one of the four possible substitutions. Thus the key for all the four S boxes (K_1, K_2) is 8 bits long and the whole key of Fig. 4.7 is $K = K_1 + K_2$ of 16 bits long.

Block ciphers, based on SPN principles, proposed by Shannon, are using products of type $S_1 P S_2 P \dots S_r$, P being a permutation without key and S_1, \dots, S_r a simple cryptographic transform based on a cryptographic key (K_r). Each iteration (round) is repeated r times using the same routine. The output of the round i is input to the round $(i+1)$. At decryption the process is the same as for encryption, but in reverse order.

In what follows, DES (Data Encryption Standard) the most studied and used until his replacement in May 2002 will be presented.

4.6.2.2 DES (Data Encryption Standard)

A very good presentation of DES history is made in [64] and samples of it will be presented in what follows, because that history represents an important step in the open / non-military / civilian cryptography development.

At the beginnings of 70th the non-military cryptography almost did not exist, despite a very good market for cryptographic products (especially governments of different outside countries), impossible to be interconnected and certified by an independent organisation.

Besides this necessity, the development of computers and data communications at large scale for civilians too, required urgently security in processing and transmission. It is true, IBM the greatest producer in the domain, had a group of valuable cryptographers who created Lucifer, a straightward algorithm for IBM, but the necessity to have a standard was imperative.

This is why NBS (National Bureau of Standards) of USA, today NIST (National Institute of Standards and Technology) initiated a programme to develop a standard for cryptographic algorithms; the requirements were to be easy to analyse and certify and the equipments using it could interoperate. It launched the public call in 1973, but the submissions were very far to the requirements.

The second call was done in 1974, when they had the promise that IBM will submit Lucifer (this one fulfilled the requirements).

NBS asked NSA (National Security Agency) to evaluate the algorithm and to determine if it is suitable as federal standard.

In March 1975, the Federal Register published the details of the algorithm and the IBM requirements for nonexclusivity and in August it requested comments on it from agencies and the general public. As a consequence, in 1976 were organised two workshops, one for specialists, to debate the mathematics and the possibility of a trap in it and the second one to discuss the problem of increasing the length of the key from 56 (NSA reduced the original length from 128 bits to 56).

This was the first and until now the last “democratic” choice of a security standard. The “voice of designers, evaluators, implementers, vendors, users and critics” [65] was heard.

Despite the criticism, especially related to the key size (56 bits plus 8 parity check), in 26.11.1976, DES was adopted as a federal standard and authorised for unclassified government communications. In 1977 was given the official description of the standard.

DES was thought by NSA for hardware implementation because of real time applications, but the published details were enough for software implementation (despite 1000 times slower [64]).

DES was the first, and until now the last cryptographic algorithm publicly evaluated by NSA, being thus an important catalyser of civilian cryptography, being available for study a secure algorithm, certified by NSA.

The main characteristics of DES are:

- block size: $n=64$
- key size: $n_k = 64 = 56 + 8$

- number of rounds: $r=16$
- SPN type

In what follows we will present DES encryption standard, more precisely one of its versions. For a better understanding of the ciphering scheme we will explain:

- round keys generation
- encryption routine
- encryption/decryption function f .

Round keys generation

The round keys are generated starting from a 64 bits key, 8 being parity-check bits. In this way the 8th bit is the parity-check bit for the first 7 bits, the 16th bit is the parity-check bit for the bits 9 to 15 bit and s. o. until the 64th bit which is the parity-check bit for the bits 57 to 63.

The key generation is presented in Fig. 4.8.

Key generation is obtained as follows:

- the key (64 bits from which 8 are parity bits) provided by the user, is delivered to the permutation box P1, which provides at the output 56 bits, and then this stream is split in two: C_0 , D_0 of 28 bits each.
- C_0 structure is:

57	49	41	33	25	17	09	01	58	50	42	34	26	18
10	02	59	51	43	35	27	19	11	03	60	52	44	36

(bit 57 from the initial matrix will be the first bit from the C_0 matrix, and bit 58 will be bit 9 from C_0 etc.)

- D_0 structure is:

63	55	47	39	31	23	15	07	62	54	46	38	30	22
14	06	61	53	45	37	29	21	13	05	28	20	12	04

(bit 63 from the initial matrix will be the first from D_0 , bit 7 will be bit 8 in D_0 etc.)

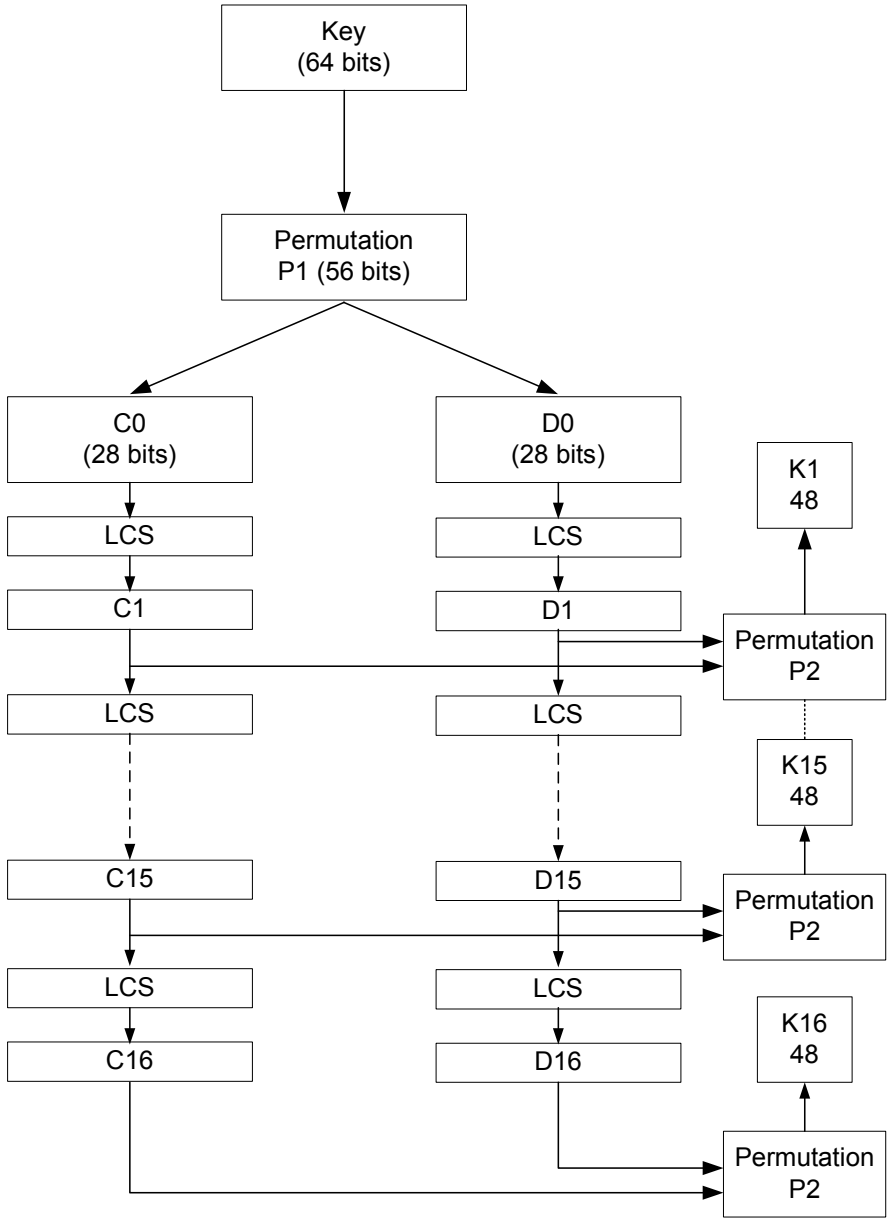


Fig. 4.8 Generation of round keys in DES

- the elements of $C_1 \dots C_{16}$ and $D_1 \dots D_{16}$ are obtained shifting to the left (LCS-left cyclic shift) the previous inputs as given below:

Round	Number of shifted bits	Round	Number of shifted bits
01	1	09	1
02	1	10	2
03	2	11	2
04	2	12	2
05	2	13	2
06	2	14	2
07	2	15	2
08	2	16	1

- the 16 keys K_i , $i = \overline{1 \dots 16}$ are obtained from matrices C_i and D_i . The two matrices C_i and D_i form a new matrix (the first 28 bits come from C_i and the next from D_i), which is introduced in a permutation box (P2) delivering at the output 48 bits with the following structure:

14	17	11	24	01	05	03	28	15	06	21	10
23	19	12	04	26	08	16	07	27	20	13	02
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

(the 14th bit of the input is the first at the output).

Encryption routine

Encryption routine is given in Fig. 4.9 and it has the following steps:

- the initial transposition matrix IP has the structure:

58	50	42	34	26	18	10	02	60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06	64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01	59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05	63	55	47	39	31	23	15	07

- L_0 contains the first 32 bits and R_0 the next 32
- XOR is a block that performs an exclusive OR between the bits of the two input blocks
- DES function f is a two variable function. Its representation is given in figure 4.1 output transposition matrix (IP^{-1}) is the inverse of IP

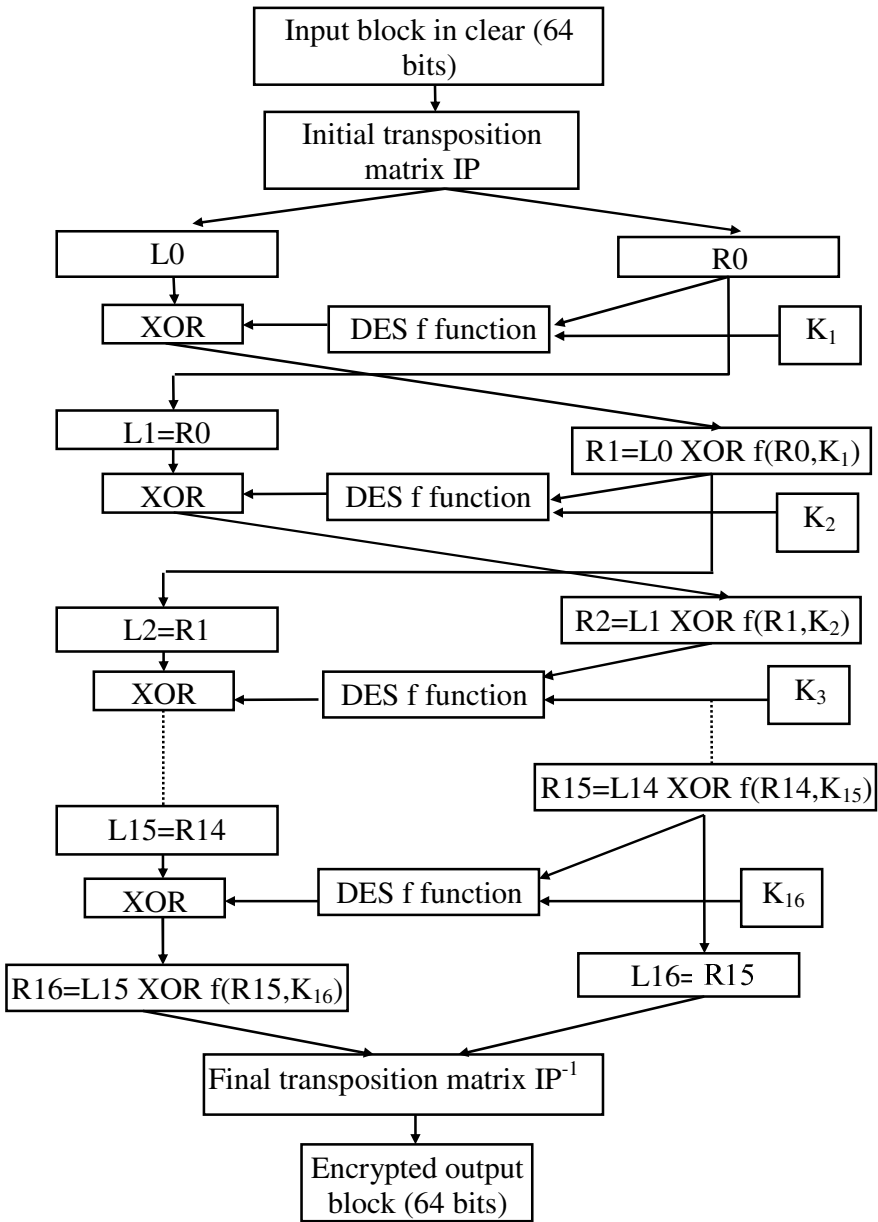


Fig. 4.9 DES encryption routine

DES encryption/decryption function f

The analysis of function f is done following Fig. 4.10:

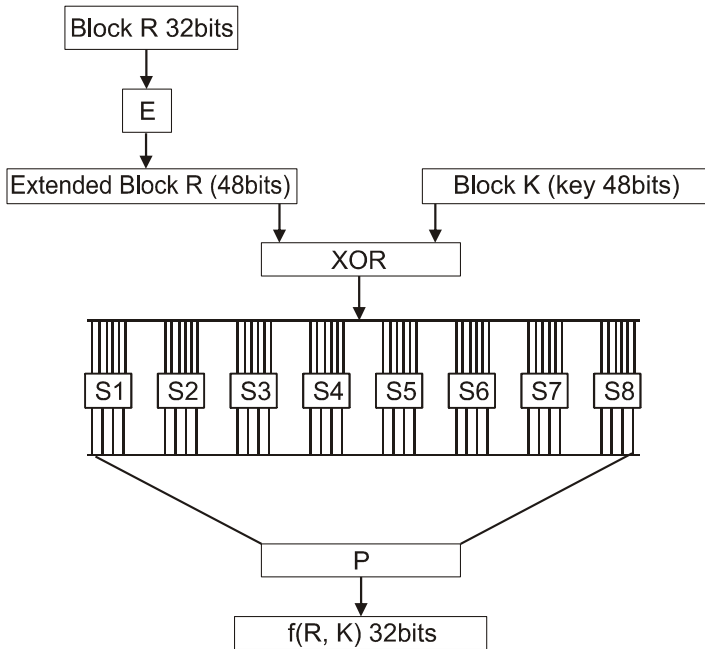


Fig. 4.10 DES encryption/decryption function f

The processing steps in function f are:

- inputs of function f are the (R_i, K_{i+1}) pairs (see fig.4.10)
- the keys K_i correspond to 48 bits blocks, whereas R_i are 32 bits blocks; in order to correlate the two blocks dimensions, an extension of R_i is performed to a 48 bits block, according to the following extension matrix E :

32	01	02	03	04	05	04	05	06	07	08	09
08	09	10	11	12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21	22	23	24	25
24	25	26	27	28	29	28	29	30	31	32	01

- the block corresponding to key K_{i+1} and the one corresponding to extended R_i are the inputs into an XOR block. The new block is divided into 8 sub-blocks of 6 bits each, sub-blocks processed through S_i blocks, $i = 1 \dots 8$. The substitution blocks S_i are distinct and correspond to the following matrices:

S1:

14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
00	15	07	04	14	02	13	01	10	06	12	11	09	05	03	08
04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

S2:

15	01	08	14	06	11	03	04	09	07	02	13	12	00	05	10
03	13	04	07	15	02	08	14	12	00	01	10	06	09	11	05
00	14	07	11	10	04	13	01	05	08	12	06	09	03	02	15
13	08	10	01	03	15	04	02	11	06	07	12	00	05	14	09

S3:

10	00	09	14	06	03	15	05	01	13	12	07	11	04	02	08
13	07	00	09	03	04	06	10	02	08	05	14	12	11	15	01
13	06	04	09	08	15	03	00	11	01	02	12	05	10	14	07
01	10	13	00	06	09	08	07	04	15	14	03	11	05	02	12

S4:

07	13	14	03	00	06	09	10	01	02	08	05	11	12	04	15
13	08	11	05	06	15	00	03	04	07	02	12	01	10	14	09
10	06	09	00	12	11	07	13	15	01	03	14	05	02	08	04
03	15	00	06	10	01	13	08	09	04	05	11	12	07	02	14

S5:

02	12	04	01	07	10	11	06	08	05	03	15	13	00	14	09
14	11	02	12	04	07	13	01	05	00	15	10	03	09	08	06
04	02	01	11	10	13	07	08	15	09	12	05	06	03	00	14
11	08	12	07	01	14	02	13	06	15	00	09	10	04	05	03

S6:

12	01	10	15	09	02	06	08	00	13	03	04	14	07	05	11
10	15	04	02	07	12	09	05	06	01	13	14	00	11	03	08
09	14	15	05	02	08	12	03	07	00	04	10	01	13	11	06
04	03	02	12	09	05	15	10	11	14	01	04	06	00	08	13

S7:

04	11	02	14	15	00	08	13	03	12	09	07	05	10	06	01
13	00	11	07	04	09	01	10	14	03	05	12	02	15	08	06
01	04	11	13	12	03	07	14	10	15	06	08	00	05	09	02
06	11	13	08	01	04	10	07	09	05	00	15	14	02	03	12

S8:

13	02	08	04	06	15	11	01	10	09	03	14	05	00	12	07
01	15	13	08	10	03	07	04	12	05	06	11	00	14	09	02
07	11	04	01	09	12	14	02	00	06	10	13	15	03	05	08
02	01	14	07	04	10	08	13	15	12	09	00	03	05	06	11

- the algorithm corresponding to a block S is the following: the first and last bit from the 6 bits input block defines the row in the matrix and the others 4 the column. The element selected this way is a number between 0 and 15. The output of block S provides the binary code of that number (4 bits).
- the output transposition matrix P is:

16	07	20	21	29	12	28	17
01	05	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

DES decrypting routine

Concerning decryption, the same algorithm is used, the only difference being that the 16 keys are used in reverse order than for encryption.

4.6.2.3 Some Other Block Ciphers

- *AES (Advanced Encryption Standard)* was selected in 26 May 2002 to replace the old standard DES.

It was the winner of NIST call for a new standard, being selected from 15 candidates.

The algorithm, designed by the Belgian cryptographers Joan Daemen and Vincent Rijmen was presented in 1998 as *Rijndael* cipher, from the names of its authors.

The main characteristics of the cipher are:

- block size: $n=128$ (2×64)
- key size: $n_k = 128$, or 192, or 256 (at choice)
- number of rounds: $r=10$, or 12, or 14 (at choice)
- SPN structure: -S boxes are based on multiplicative inverse over $GF(2^8)$ (see finite fields in Annex A) with good non-linearity properties; P boxes are based on permutations of lines and columns, treated as polynomial in $GF(2^8)$.

The USA government certified in June 2003 that AES can be used for classified information of level SECRET (with key of size 128) and TOP SECRET (with keys of size 192 and 256).

Soon after becoming the official standard for symmetric cryptography, were reported attacks on AES: at round 7 for key size 128, at round 8 for key size 192 and round 9 for key size 256 (it looks that strong debates as those made for DES, are important despite our rushing times).

- *IDEA (International Data Encryption Algorithm)*, invented in 1990 by X. Lai and J. Massay in Switzerland, is a very secure, flexible and quick cipher [64]; it was one of the most promising candidates for the new standard. Its main features are:

- block size: $n=64$ (as for DES)
- key size: $n_k = 128$
- number of rounds: $r=8$ (less than DES which has 16, meaning that is much faster)
- flexible and cheap software implementation and very fast operation time in hardware implementation using FPGA facilities
- the principle is based on modulo 2^{16} summation and modulo $(2^{16} + 1)$ multiplication, operations which are not distributive and associative, making thus difficult the cryptanalysis.

Despite IDEA lost the battle for the new standard, it remains a very secure, fast, flexible and cheap cipher, being widely used in PGP (Pretty Good Privacy) [64].

- *Multiple DES* [64], [68]

The time when DES become wick because of his small size key (56 bits), alternatives were investigated, in order to preserve the existing equipment, but to enhance the security. A solution was to use multiple encryption with DES and multiple keys. The most common are: Double- DES with two keys, using an equivalent key size of $56 \times 2 = 112$ bits and Triple- DES (T- DES) with two or three keys and an equivalent key size of $56 \times 3 = 168$ bits.

Applications of T- DES with two keys: PEM (Privacy Enhanced Mail), ANSI (American National Standard Institute) and ISO 8732 for key generation.

Applications of T- DES with three keys are in PGP and S/MIME (Secure/ Multiple purpose Internet Mail Extension).

DES-X [64] is a DES improvement proposed by Ron Rivest from RSA Data Security; it uses the *whitening technique*, a cheap way to improve the security hiding the input and the output of a cryptosystem (Fig. 4.11).

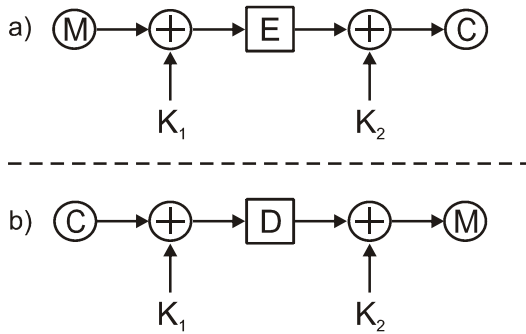


Fig. 4.11 Illustration of whitening technique: a- encryption, b- decryption

Remark

In practice: $K_1 = K_2 = K$.

- *CAST- 128* or *256* is a symmetric block cipher used in Canada and invented in 1997 by Carlisle Adams and Stafford Tavares. It was between the 15 candidates for the selection of the new standard. The main features are:

- block size: $n=64$ bits
- key size is variable, being a multiple of 8 : $40 \div 256$ bits
- number of rounds: $r=16$

- *SAFER* [64] (Secure and Fast Encryption Routine) invented by J. Massay, was a candidate too for AES. Its main characteristics are:

- block size: $n=128$ bits
- key size is variable: 128, 192 or 256 bits
- is based on modular arithmetic.

- *FEAL* [64] is the equivalent DES of Japan, having almost the same characteristics as size:

- block size: $n=64$ bits
- key size: $n_k = 64$ bits

but operating in modular arithmetic: $\text{mod} (2^8)$.

- *GOST* (Gosudarstvenñi Standard) is the governmental standard of Russia. It was adopted in 1989 as an equivalent for DES. Its characteristics are:

- block size: $n=64$ bits (as DES)
- key size: $n_k = 256$ (much greater than DES- 56, and obviously longer life)

- number of rounds: $r=32$ (greater than DES- 16)
- the S and P boxes are secret, despite Kerckhoffs law.

4.6.2.4 Block Cipher Operation Modes

An *operation mode* of a block cipher is defined by the used algorithm (cipher), the feedback and some elementary transformations. The most used algorithm is DES, but the modes act the same for any block cipher.

Basically there are four operation modes.

1. *Electronic Codebook* (ECB)

Each plaintext block M_i is encoded independently using the same key K and giving the ciphertext C_i (Fig. 4.12). Because the transformation $M_i \leftrightarrow C_i$ is bi-univoc it is theoretically possible to create a code book of plaintexts M_i and the correspondent ciphertexts C_i . If the size of the block is n , there will be 2^n distinct M_i ; if n is great, this code book will be much too large to be stored and further, for each key a distinct code book is obtained.

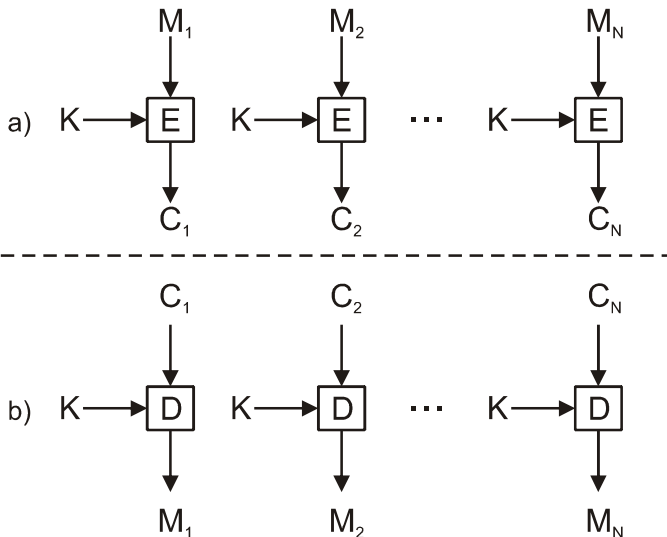


Fig. 4.12 ECB mode: a) encryption; b) decryption

Advantages

- the encryption of blocks being independent, is suitable for randomly accessed encrypted files (as an encrypted database); any record could be added, deleted, encrypted or decrypted independently [64].
- the process suit to parallelization, using multiple processors for encryption and decryption of distinct blocks.
- each block is independently affected by errors

Drawbacks

- if the block M_i is repeated, the ciphertext C_i is repeating too, giving thus information to the cryptanalyst. Ways to avoid this is to have distinct block M_i (to avoid stereotyped beginnings and endings).
- padding occurs when the block is smaller than n .
- the most serious problem in ECB is when an adversary could modify the encrypted files (*block reply* [64]); to avoid this situation, block chaining is required.

2. *Cipher Block Chaining (CBC)(Fig. 4.13)*

Is used precisely for long plaintext ($>n$), where there are blocks M_i which are repeated. In order to obtain distinct C_i for identical M_i , a feedback is introduced, such that the input of the encryption block will be the current plaintext M_i added modulo two with the ciphertext of the previous encryption block (Fig. 4.13) $M_i \oplus C_{i-1} = C_i \neq C_{i-1}$ if $M_i = M_{i-1}$.

Advantages

- CBC mode, in order to have distinct ciphertexts C_i for identical M_i , starting from the first block, is using in the first block an *initialization vector (IV)*, which plays also an *authentication role*, but it can be transmitted in clear, only the key k requiring secure channel in transmission.

Drawbacks

- *error propagation*: errors in the ciphertext propagate, affecting more blocks in the recovered plaintext (*error extension*).
- security problems, such birthday attack is discussed in [64].
- padding remains a problem as in ECB.

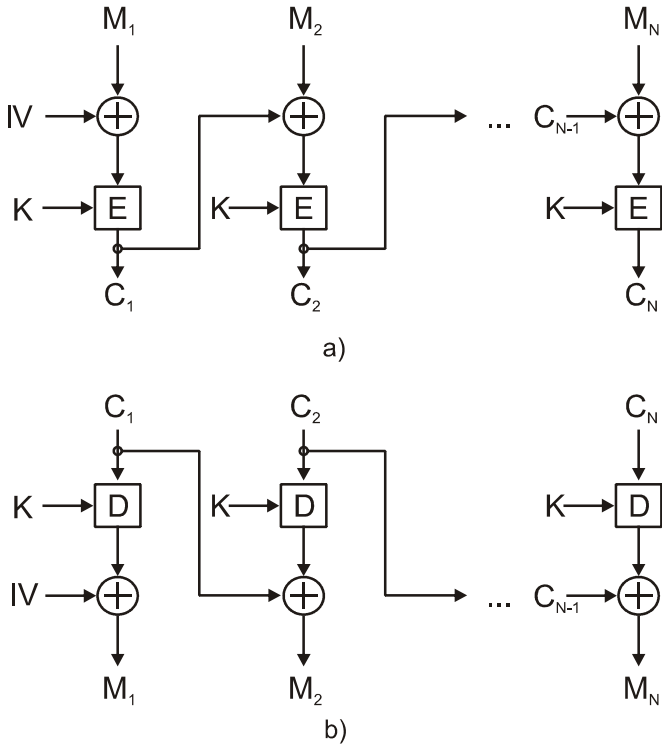


Fig. 4.13 CBC mode : a) encryption, b) decryption

3. *Cipher Feedback (CFB)*

CFB is used to transform a block cipher into a stream cipher. The encryption is done on j bits at once ($j=1, 8, 32$), thus the size of M_i being also j . It requires in the first block an initialization vector (IV), as in CBC (Fig. 4.14). Decryption follows the same steps, but in reverse order.

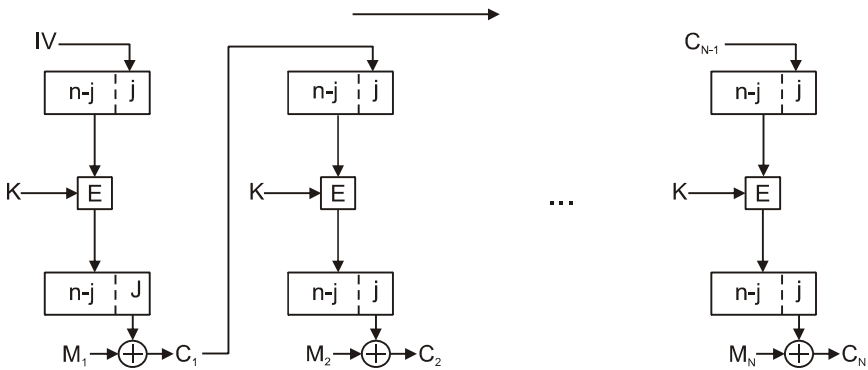


Fig. 4.14 CFB mode: encryption

As stated in previous chain modes, the existence of an identification vector (IV), in the first block, is used also for authentication.

4. Output Feedback (OFB)

OFB is identical with CFB as structure (Fig. 4.15), the difference being that for chaining are used the j bits and not C_1 as in CFB, which ensure that errors in transmission do not propagate. Decryption follows the same steps, but in reverse order.

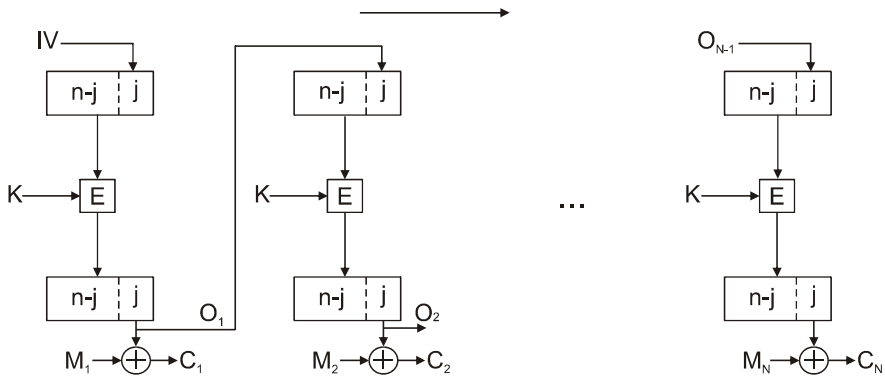


Fig. 4.15 OFB mode : encryption

4.6.3 Stream Ciphers

4.6.3.1 General Features

Stream ciphers are modern versions of Vigenère cipher, using as basic transformation substitutions, to make confusion.

- They process the information continuously bit by bit or character by character.
- Are fast in hardware implementation (using linear feedback shift registers – LFSR, see chapter 5).
- Are used in noisy applications, based on their advantage of do not propagate the errors, such radio channels.
- They can simulate OTP principles, meaning the generation of long pseudo-random keys in two ways:

1. Linear congruential generators

A linear congruential generator is a generator of a pseudo- random (noise) sequence.

$$X_n = (aX_{n-1} + b) \bmod m \quad (4.15)$$

where X_n is the n-th term of the sequence, X_{n-1} the (n-1)-th term, and

- a- multiplier
- b- increment
- m- modulus

The seed (key) is the value X_0 .

The sequence generated by relation (4.15) is a sequence with a period no greater than m, which is the maximal period and is obtained if b is relatively prime to m. Details concerning such generators are given in [64].

2. *LFSR as pseudo- random (noise) generators*

A pseudo-random sequence can be generated using a linear feedback shift register (LFSR) having the connections in correspondence to the coefficients of primitive polynomials $g(x)$, such that generated sequence length is maximum (see also 5.8.5).

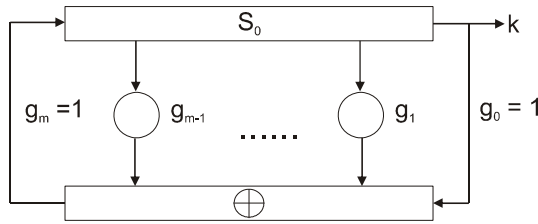


Fig. 4.16 Block scheme of a pseudo-noise generator using LFSR

The sequence at the output, k , of the register will be:

$$k = [a_0 a_1 \dots a_{n-1}], \quad n = 2^m - 1 \tag{4.25}$$

Each symbol a_i is given by:

$$a_i = \sigma T^i S_0 \tag{4.26}$$

where

- σ is the output selection matrix of the pseudo-random sequence (the output can be taken from any of the m LFSR cells); in Fig. 4.16 the expression of σ is:

$$\begin{matrix} \sigma = [10\dots 0] \\ \uparrow \\ \text{LSB} \end{matrix} \tag{4.27}$$

- T is the register characteristic matrix
- $g(x) = g_m x^m + g_{m-1} x^{m-1} + \dots + g_1 x + g_0$ is LFSR generator polynomial
- S_0 is initial state of LFSR; obviously, it cannot be zero

Changing the coefficients of the generator polynomial and the initial state of the shift register, we can modify the generated sequence and subsequently the keys k .

At the receiver, an identical generator and synchronised with the one used at emission, generates the same pseudo-random sequence k , which added modulo 2 with the received sequence gives the message in clear.

Example 4.8

Let us consider a pseudo-random sequence generator with the generator polynomial: $g(x) = x^3 + x^2 + 1$. The block scheme of this generator is presented in Fig. 4.17.

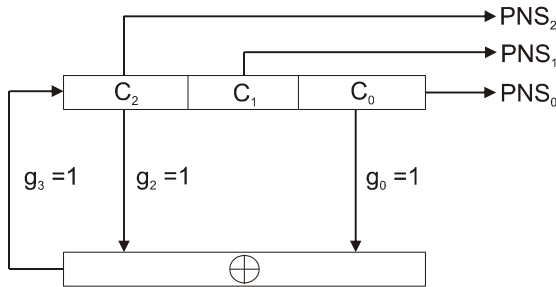


Fig. 4.17 Pseudo-noise generator with LFSR and $g(x)=x^3 + x^2 + 1$

Considering the shift register initial state as 101, the pseudo-random sequence can be obtained by register time evolution:

Clock t_n	SR t_{n+1}			Pseudo-noise sequence (PNS) t_n		
	C_2	C_1	C_0	PNS_2	PNS_1	PNS_0
1	1	0	1	1	0	1
2	0	1	0	0	1	0
3	0	0	1	0	0	1
4	1	0	0	1	0	0
5	1	1	0	1	1	0
6	1	1	1	1	1	1
7	0	1	1	0	1	1

Remarks

- at the 8th clock, the register will contain the initial sequence 101, so the process will be repeated
- the pseudo-noise sequence is 7 bits long: $n=2^{m-1}=7$

Longer keys can be obtained easily multiplexing two or more LFSR of small lengths (Fig. 4.18).

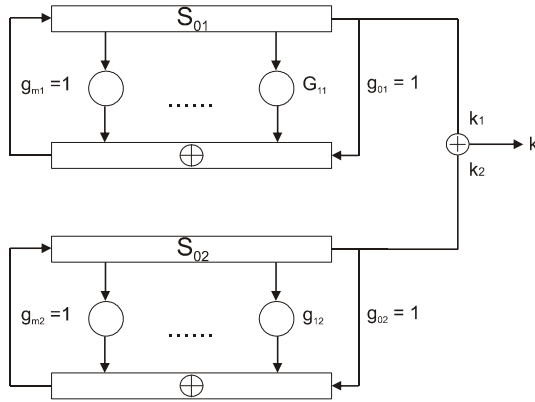


Fig. 4.18 Pseudo noise sequences generator implemented with two LFSRs

According to (4.26), any element a_i of the pseudo noise sequence k , will be:

$$a_i = \sigma_1 T_1^i S_{01} + \sigma_2 T_2^i S_{02} \tag{4.28}$$

If n is the length of the k -th generated sequence (the period), then:

$$\sigma_1 T_1^{i+n} S_{01} + \sigma_2 T_2^{i+n} S_{02} = \sigma_1 T_1^i S_{01} + \sigma_2 T_2^i S_{02} \tag{4.29}$$

$$T_1^{n_1} = I_1, \quad T_2^{n_2} = I_2 \tag{4.30}$$

$$n_1 = 2^{m_1} - 1, \quad n_2 = 2^{m_2} - 1 \tag{4.31}$$

In order to fulfil relation (4.28) we need to have:

$$T_1^n = I_1 \quad \text{and} \quad T_2^n = I_2 \tag{4.32}$$

The smallest n that satisfies this condition is the least common multiple of numbers n_1 and n_2 :

$$n = \text{l.c.m.}\{n_1; n_2\} \tag{4.33}$$

If n_1 and n_2 are prime numbers:

$$n = n_1 \cdot n_2 \tag{4.33.a}$$

The difficulty of the cryptanalysis is increased if a certain non-linearity is introduced (Fig. 4.19).

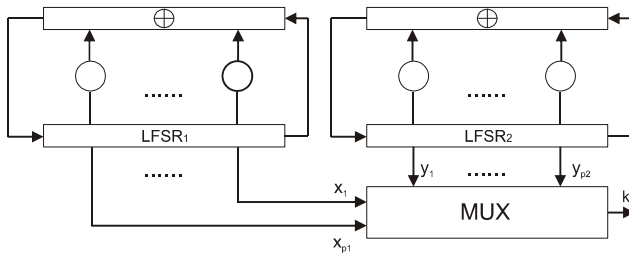


Fig. 4.19 Nonlinear multiplexed system for generating pseudo-noise sequences

Example 4.9

For the particular case when $p_1 = 3$ and $p_2 = 8$ the output k of the multiplexing circuit is expressed, underlining the non-linearity introduced by multiplexing.

x_3	x_2	x_1	Y
0	0	0	y_1
0	0	1	y_2
0	1	0	y_3
0	1	1	y_4
1	0	0	y_5
1	0	1	y_6
1	1	0	y_7
1	1	1	y_8

$$k = \bar{x}_1\bar{x}_2\bar{x}_3y_1 + x_1\bar{x}_2\bar{x}_3y_2 + \bar{x}_1x_2\bar{x}_3y_3 + x_1x_2\bar{x}_3y_4 + \bar{x}_1\bar{x}_2x_3y_5 + x_1\bar{x}_2x_3y_6 + \bar{x}_1x_2x_3y_7 + x_1x_2x_3y_8$$

4.6.3.2 Some Stream Ciphers

A wider presentation of stream ciphers is given in [64], [68]. Some of them will be mentioned in what follows:

A5 is used in GSM (Group Special Mobile) to encrypt the link between base station and subscribers. It was designed by French cryptographers in 1980 and contains three LFSRs with $m_1 = 19$, $m_2 = 22$, $m_3 = 23$. In this way, using the principle described above, the length of the key, obtained by multiplexing (relation (4.33.a)) is:

$$n = n_1 \cdot n_2 \cdot n_3,$$

where $n_1 = 2^{19} - 1$, $n_2 = 2^{22} - 1$, $n_3 = 2^{23} - 1$.

The security is medium, but it is quick, error resistant and chip.

RC-4 was invented by Ron Rivest in 1987. It is a block cipher operating in OFB mode. In 1994 its source code was put on the internet, and for this reason it is extremely studied and used.

SEAL is a stream cipher created at IBM, very fast in software implementation.

Hughes XPD/ KPD was created in 1986 to protect portable devices (XPD) and is used by Hughes Aircraft Corporation for kinetic protection device (KPD). It is basically composed by a single LFSR of 61 bits long ($m=61$); at this degree, the number of primitive polynomials (see Appendix A) is great: 2^{10} , meaning that the key is composed of two subkeys the choice of the characteristic polynomial (one from 2^{10} possible) and the initial state S_0 of the LFSR (there are $(2^{61} - 1)$).

NANOTWO was used by the South African police for fax transmission protection. The principle is similar to the one used in Hughes XPD/ KPD: a unique LFSR 127 bits long, meaning that S_0 and $g(x)$ are selected from greater sets.

4.6.4 Authentication with Symmetric Cryptography

In networks with several users and several data bases, a user must be convinced that he entered into the desired data-base, and the computer (database) has to be sure that that person is authorised to have access to these databases. In this case, between user A and computer C an authentication protocol is established:

- the user gives his clear identity A and randomly composes a short data sequence X which is ciphered with his own key (k_A):

$$(X \oplus k_A) \quad (4.34)$$

- the computer knows the key corresponding to A and decipheres the sequence X with the same key (k_A):

$$(X \oplus k_A) \oplus k_A = X \quad (4.35)$$

- to this deciphered sequence, the computer adds its own random sequence Y and ciphers both sequences with key k_A , transmitting the sequences to A:

$$(XY) \oplus k_A \quad (4.36)$$

- when deciphering the message, A compares the received sequence X to the transmitted sequence X and makes sure of computer identity data (authentication):

$$[(XY) \oplus k_A] \oplus k_A = XY \quad (4.37)$$

- in the next messages transmitted by A, it adds the deciphered sequence Y, ciphers everything with its own key and transmits it to the computer:

$$(MY) \oplus k_A \quad (4.38)$$

- the computer compares the deciphered sequence Y with the sequence sent for the first time and makes sure that the correspondent is the right one (entity authentication):

$$(MY) \oplus k_A \oplus k_A = MY \quad (4.39)$$

4.7 Public Key Cryptography

4.7.1 Principle of the Public Key Cryptography

The year 1976 is considered the starting point of modern cryptography. In November 1976, W. Diffie and M. Hellman published in IEEE Transaction on Information Theory the paper: “New Direction in Cryptography” in which they introduced the PKC (Public Key Cryptography) concept.

The principle of this concept is shown in fig. 4.20:

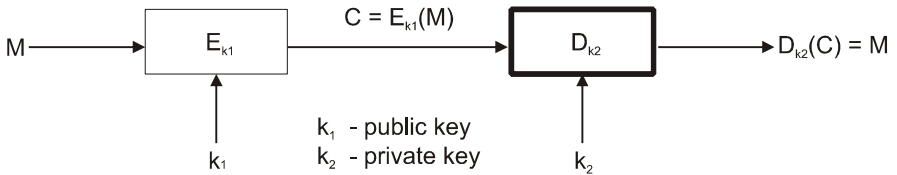


Fig. 4.20 Block scheme of a public key system

Each user has for encryption a public transform E_{k_1} , that can be memorized in a public folder and a transform for secret decryption D_{k_2} .

The *requirements* of public key cryptography are:

- for each pair of keys (k_1, k_2) , the decryption function with key k_2 : D_{k_2} is the reverse of the encryption function with key k_1 : E_{k_1}
- for any pair (k_1, k_2) and any M, the computation algorithms for E_{k_1} and D_{k_2} are easy and fast
- for any pair (k_1, k_2) , the computation algorithm for D_{k_2} cannot be obtained in reasonable time starting from E_{k_1} ; decryption D_{k_2} (which is secret) is derived from E_{k_1} through a transformation that cannot be easily reversed (one way function)
- any pair (k_1, k_2) must be easily obtained starting from a unique and secret key

In public key systems, confidentiality and authentication are done by distinct transform.

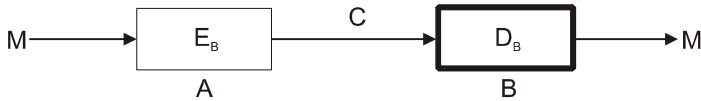


Fig. 4.21 Confidentiality in public key cryptosystems

Let us assume that user A wants to transmit a message M to another user B. In this case, A, knowing the public transform of B (E_B), transmits the following ciphertext:

$$C = E_B(M) \quad (4.40)$$

ensuring confidentiality (Fig 4.21)

When receiving the message, B decrypts the ciphertext C using the secret transform D_B :

$$D_B(C) = D_B(E_B(M)) = M \quad (4.41)$$

Such a system does not allow any authentication because any user has access to the public transform of B (E_B) and can transmit a fake messages (M'):

$$C' = E_B(M') \quad (4.42)$$

For *authentication* (Fig.4.22) we apply to M the secret transform D_A of A. A will transmitte to B:

$$C = D_A(M) \quad (4.43)$$

When receives the encrypted message, B will apply the public transform E_A corresponding to A:

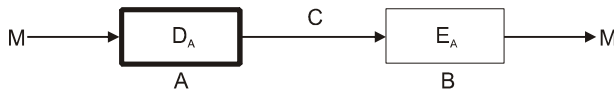


Fig. 4.22 Authentication in public key cryptosystems

Authentication is done, because fake messages cannot be transmitted to B: $C' = D_A(M')$, only A knows D_A (the secret key). In this case, however, the confidentiality is not ensured, due to the fact that M can be obtained by anyone just applying E_A to C , E_A being public.

In order to simultaneously ensure *confidentiality and authentication* (Fig 4.23), the space M must be equivalent to the space C , so that any pair (E_A, D_A) , (E_B, D_B) can operate not only on clear text but also on ciphered one. It is necessary for (E_A, D_A) and (E_B, D_B) to be mutually inverse:

$$E_A(D_A(M)) = D_A(E_A(M)) = M \quad (4.44)$$

$$E_B(D_B(M)) = D_B(E_B(M)) = M \quad (4.45)$$

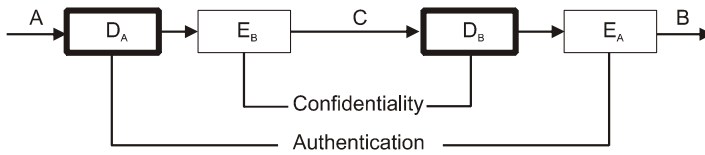


Fig. 4.23 Confidentiality and authentication in public key cryptosystems

User A will sign using his private key D_A the message M and then will encrypt it using the public key of B:

$$C = E_B(D_A(M)) \quad (4.46)$$

B will obtain M applying to the received cipher text its own private key D_B and then the public key of A (E_A) to check his signature:

$$E_A(D_B(C)) = E_A(D_B(E_B(D_A(M)))) = E_A(D_A(M)) = M \quad (4.47)$$

Digital signature in public key cryptosystems

Let us consider that B is receiving a signed message from A. The signature of A needs to have the following properties:

- B is able to validate A signature
- it must be impossible for anyone, including B, to fake A signature
- when A does not recognize the message M signature, it has to be a “judge” (Trusted Person - TP) to solve the argue between A and B

The implementation of digital signature is very simple in public key systems. In this case D_A can serve as a digital signature for A. The receiver B of the message M signed by A knows that transmitter and data are authentic. Because E_A transform is public, the receiver B can validate the signature.

The *protocol of digital signature* can be developed as follows:

- A signes M:

$$S = D_A(M) \quad (4.48)$$

- A sends to B the cryptogram:

$$C = E_B(S) \quad (4.49)$$

- B validates A signature, verifying if

$$E_A(S) = M \quad (4.50)$$

$$(D_B(C)) = D_B(E_B(S)) = S \quad (4.51)$$

$$E_A(S) = E_A(D_A(M)) = M \quad (4.52)$$

- a “judge”(TP) solves the dispute that may occur between A and B checking if $E_A(S)$ belongs to M, in the same manner as B does.

4.7.2 Public Keys Ciphers

Rivest – Shamir - Adleman (RSA) cipher

This algorithm was published in 1978. The security of the cipher is based on the fact that the factorisation of the product of two high prime numbers is, at least for now, unsolvable. Related to this aspect, Fermat and Legendre developed some factorisation algorithms. The most efficient ones, used even now, are the ones developed by Legendre[96].

Encryption method implies exponential computation in a Galois Field (GF(n)). Ciphertext can be obtained from the cleartext through a block transform (encoding). Be M such a cleartext block, having the property $M \in (0, n-1)$. The encrypted block C corresponding to the cleartext block can be obtained computing the exponential: $C = M^E \pmod{n}$, E and n representing the public key. Decryption is done as: $M = C^D \pmod{n}$, D being the secret decryption key (private key).

The two keys, E and D, must satisfy the following relations:

$$M = C^D \pmod{n} = M^{ED} \pmod{n} \quad (4.53)$$

This algorithm is based on *Euler-Fermat Theorem*: if n is prime number and a a positive integer, non divisible, with n, then $a^{n-1} = 1 \pmod{n}$ for any $a \in [1, n)$.

If we chose n a prime number, for any block $M \in (0, n-1)$ we have:

$$M^{\varphi(n)} \pmod{n} = 1 \quad (4.54)$$

where

$$\varphi(n) = n-1 \quad (4.55)$$

is *Euler totient*.

If E and D satisfy the relation:

$$ED \pmod{\varphi(n)} = 1 \quad (4.56)$$

then we may write:

$$ED = k\varphi(n) + 1 = \varphi(n) + \varphi(n) + \dots + \varphi(n) + 1 \quad (4.57)$$

$$M^{ED} = M^{\varphi(n)+\varphi(n)+\dots+\varphi(n)+1} = M^{\varphi(n)} \cdot M^{\varphi(n)} \dots M \pmod{n} \quad (4.58)$$

It follows:

$$ED(\text{mod } \varphi(n)) = 1 \Rightarrow M^{ED} = M(\text{mod } n) \quad (4.59)$$

In this way we made a reversible transform based on exponential finite fields. We still have to solve the problem concerning the security of the decryption key. This key D must be almost impossible to be determined from the encryption key, but in the given case it is easy to determine this key using E and n and knowing that $ED(\text{mod } \varphi(n)) = 1$ and $\varphi(n) = n - 1$.

The security is based on high number factorization. Starting from this idea the number n can be obtained from the product of two high prime numbers p and q : $n = p \cdot q$, such that Euler totient, in this case:

$$\varphi(n) = (p - 1)(q - 1), \quad (4.60)$$

becomes harder to be found using only n .

Using this method we can obtain a secure public key system. Such a system which ensures confidentiality has as elements the following pairs:

(E, n) the public key
(D, n) the private key

A cryptanalyst who knows the pair (E, n) must determine D taking into account that $ED(\text{mod } \varphi(n)) = 1$. For this purpose he has to determine $\varphi(n)$ from $\varphi(n) = (p - 1)(q - 1)$, *i.e.* p and q ; this problem reduces to a high number factorization and it is very difficult to be solved.

Example 4.10

We choose two prime numbers: $p = 47$ and $q = 97$.

$$n = p \cdot q = 3713$$

Choosing $D = 97$, E will be 37, to satisfy:

$$D(\text{mod } (p - 1)(q - 1)) = 1$$

$$E = [(p - 1)(q - 1) + 1] / D$$

In order to encode the message "A SOSIT TIMPUL", first we must transform the letters of the alphabet in numbers. For example $A = 00$, $B = 01$,

The message becomes:

001814180819190802152011

In what follows we encode each 4 numbers smaller than n :

$$0018^E \text{ mod}(n) = 0018^{37} \text{ mod}(3713) = 3019$$

$$1418^E \text{ mod}(n) = 1418^{37} \text{ mod}(3713) = 0943$$

The encrypted message becomes:

309109433366254501072965

At the decryption we compute:

$$3091^D \bmod(n) = 3091^{97} \bmod(3713) = 0018$$

thus obtaining the original message.

Once the method is understood, it may easily hint its broad field of applications in authentication and digital signatures.

However a problem that still may appear in developing such an algorithm is computing systems values: the number n and the two keys E and D ; the computation is done at tens digits level in order to assume a high level of security. Until now lengths of (512 – 1024) bits are considered enough for current applications.

The encryption system with public keys RSA is the most important among public key algorithms, offering a high level of security and being a standard in digital signatures field. RSA is known as the safest PKC algorithm of encryption and authentication commercially available, impossible to be broken even by governmental agencies. The method has a great advantage because, compared to some other encryption methods, there are no traps for system breaking. The algorithm is used for confidentiality and data authentication, passwords, being used by a lot of companies as: DEC, Lotus, Novell, Motorola, and also a series of important institutions (the USA Defence Department, Boeing, the bank network SWIFT - Society for Worldwide Interbank Financial Telecommunication, the Belgium Government, etc).

4.8 Digital Watermarking

4.8.1 Introduction

In the last 20 years we were witnesses of an outbreak of the digital multimedia technologies. The digital audio/ video information has several advantages over its analogical counterpart:

- superior quality in transmission, processing and storage
- simpler editing facilities, the desired fragments of the initial data can be located with precision and modified
- simpler lossless copying : the copy of a digital document is identical with the original.

For producers and distributors of multimedia products, several of the above mentioned advantages are handicaps, leading to important financial losses. Unauthorized copying of CDs, DVDs is currently a major problem. Also the information

contained in WebPages, books, and the broadcasted information, are frequently copied and used without any permission from the “editor”.

The *copyright* in this domain is a problem of maximum urgency. Several attempts in this sense exist, but we cannot speak of a generalized corresponding legislation. In 28 Oct. 1998, the president of the United States signed an act [79] (*Digital Millennium Copyright Act*) that contains recommendations to be followed in order to protect the intellectual property and also the customers rights. At its turn, the European Community is preparing several protection measures for digital multimedia products such as CDs and DVDs.

The most important technologies used in copyright protection for authors or distributors are: *encryption and watermarking*.

Encryption is used for protecting data in transmission and storage. Once the information was decrypted, it is no longer protected and can be copied without any restriction.

Watermarking is an operation, which consists in embedding an imperceptible signal called *watermark* (WM) into the host information. The host information can be text, audio signal, still image or video.

The name watermark comes from the words “water” and “mark” and designates a transparent, invisible mark like the water transparency.

In general, the watermark contains information about the origin and destination of the host information. Even though it is not directly used in intellectual property protection, it helps identifying the host and the receiver, being useful in disputes over authors / distributors rights.

From a theoretical point of view the watermark has to permanently protect the information, so it has to be robust, in such a way that any unauthorized removal will automatically lead to quality degradation. The watermark resembles to a signature, at the beginning it was called *signature*, but in order to eliminate the confusions with the digital signatures from cryptography the original name was dropped. Taking into account the fact that it has to be transparent, imperceptible for hearing or seeing, the resemblance with the “invisible tattoo”, made by A. Tewfik [76], is suggestive.

In order to insure copyright protection, the watermarking technologies need two operations (Fig. 4.24):

- watermark insertion in host data, before transmission or storage;
- watermark extraction from the received data and comparison between the original watermark and the extracted one, in case of dispute.
- Watermarking is used especially for information protection such as:
- *Copyright protection*. The owner inserts a watermark containing information related to its intellectual rights. The watermark resembles to ISBN (*International Standard Book Numbering*) - 10 characters or ISRC (*International Standard Recording Code*) - 12 alphanumerical characters.

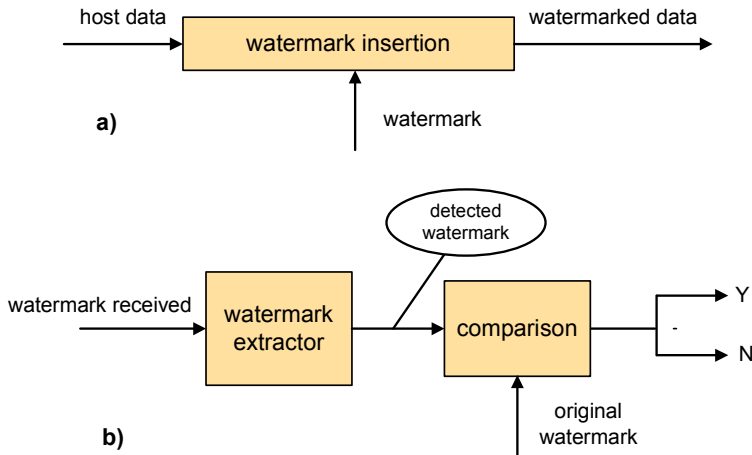


Fig. 4.24 Watermark principle block scheme: a) insertion (embedding), b) retrieval / detection

The information inserted could be related to license rights, distribution agreements, etc., in these cases watermark length being usually $60 \div 70$ bits.

- *Copy protection*; in this case the watermark is a single bit that allows (forbids) copying; this bit is computed in watermark detectors in storage devices and accordingly, information copying will be allowed (forbidden) [47].
- *Fingerprinting*, used for unauthorized copy detection. The data owner inserts information related to the customers that bought the license in the watermark. This information is like a serial number. When illegal copies are found, the source can be easily identified using the information embedded into the watermark.
- *Broadcast monitoring*: using watermarking on commercials, a monitoring system for commercials broadcasting according to the license agreements can be implemented.
- *Data authentication*: when the watermark is used for identification, it is called *fragile watermark* and it shows if the data have been altered, together with the place where the modification was done [87].

Beside these protection goals, watermarking can be used also for:

- *Characteristic enrichment for the host signal*, e.g. several language subtitling; there are several services that use this property.
- *Medicine applications*: using watermarking techniques, patient data are inserted into the medical images.
- *Secret message transmission*: there are countries where cryptographically services are restricted; it follows that secret (private) messages can be inserted through watermarking.

4.8.2 Short History

Nowadays *digital watermarking* is a modern version of *steganography* (from the Greek words “stegano” which means covered and “graphos” meaning writing) - signifying covered writing.

Steganography is a technique used for secret message hiding into other messages in such a way that the existence of the secret messages is hidden. The sender writes a secret message and hides it into an inoffensive one.

Among the techniques used during the history of steganography we remind:

- use of invisible inks
- thin holes for some characters, fine modifications of the space between words
- the use of semagrams (from the Greek words “sema” meaning sign and “gramma” meaning writing, drawing).

These techniques were recently resumed and put into digital context for text watermarking [20].

Audio watermarking (*audiosteganography*) and still / dynamic image watermarking (*videosteganography*) are using the same ideas as steganography.

As an example for *audiosteganography*, we can remind Bach. He used invisible watermarks for copyright protection, writing his name in several works using invisible watermarking; for example he counted the number of appearances of a musical note (one appearance for A, two for B, three for C and eight for H).

As for steganography, for graphical images for instance, using the least significant bit, several secret messages can be hidden. The image rests almost the same and the secret message can be extracted at the receiver. Proceeding like that for a 1024x1024 black and white image one can insert 64 KB of secret messages (several modern services are using this capacity).

For digital imaging, the first invisible marks were used in 1990 in Japan [73] and independently, in 1993 in Europe [23] and [77]. At the beginning the terminology used for such invisible marks was “label” or “signature”; around 1993 the words *water mark* were used, signifying a transparent, invisible mark. The combination of the two words, gave the word “*watermark*”, which will be used henceforward.

Applications of digital watermarking for audio domain are known since 1996 [11].

In 1995 the first applications for uncompressed and compressed still images are done [24].

1996 [33], 1997 [46] are marking the beginning for uncompressed, respectively compressed video signals.

After several breakthroughs between 1995 and 1998 it seems that the last years can be viewed as a plateau in watermarking research. Simultaneously the industry had an increasing role in standards and recommendations elaboration. This phenomenon resembles to the development of modern cryptography and the elaboration of standards for civil applications.

4.8.3 Watermarking Requirements

Each watermarking application has specific demands. However there are some general, intuitive requirements.

- *Perceptual transparency.* It is related to the fact that the watermark insertion must not affect the quality of the host data. The mark is invisible if one cannot distinguish between the original signal and the marked one, e.g. if the changes in the data are below the thresholds of human senses (hearing, seeing). Perceptual transparency test are made without knowing the input data. Original or marked data are presented independently to the subjects. If the selection percentage is equal for the two cases, this means that perceptual transparency is achieved. In real perceptual transparency applications, the subjects do not know the original data, having therefore correct testing conditions.
- *Robustness* is the watermark property to resist to unintentional changes, due to the inherent processing related to the transmission / storage (unintentional attacks) or to intentional changes (intentional attacks) aiming to remove the watermark.

There are some applications when robustness is not required. For data authentication for example, the fragile watermark needs not to be robust, an impossible watermark detection proving the fact that the data is altered, being no longer authentic.

However, for most applications the watermark has to be robust, its extraction from the host data leading to a significant quality loss, making the host data unusable.

- *Watermark payload;* the watermark payload is also known as *watermark information*. The watermark payload is defined as the information quantity included in the watermark. It is application dependent [47] and some usual values are:
 - 1 bit for copy protection
 - 20 bits for audio signals
 - $60 \div 70$ bits for video signals

Another important parameter related to the payload is the *watermark granularity*. This parameter shows the required quantity of data necessary for the insertion of a single watermark information unit. In the above-mentioned example a watermark information unit has 20 bits for audio signals and, $60 \div 70$ bits for video signals. These bits are inserted in 1 or 5 seconds for audio segments. For video signals the watermark information unit is inserted in a single frame or is spread over multiple frames.

Watermark spreading improves the detection robustness [33]. For most video applications, the watermark information is inserted in less then a second for video signals (approx. 25 frames).

- *Detection with and without original signal.* Depending on the presence of the original signal there are two methods for watermark detection [47]:
 - with the presence of the original signal : nonoblivious (informed) watermarking
 - without original signal: oblivious (public, blind) watermarking.

The first type of detection that needs the original signal or a copy of it is used in copyright protection applications restraining the inversion attack [25], [84].

The second detection modality, not needing the original, is used in application where the presence of the original at detection is impossible, for example in copy protection.

- *Security in watermarking* can be seen as in cryptography: contained in the encryption key. Consequently the watermarking is robust if some unauthorized person cannot eliminate the watermark although this person knows the insertion and detection algorithm. Subsequently the watermark insertion process uses one or several cryptographic robust keys. The keys are used also in the watermark detection process. There are applications, like covered communications, where encryption is necessary before marking.
- *Ownership deadlock.* The *ownership deadlock* is known as the inversion attack, or IBM attack, [10]. Such an attack appears whenever in the same data there are several watermarks claiming the same copyright. Someone can easily insert his own watermark in the data already marked.

Watermarking schemes capable of solving this problem (who is the “right” owner or who was the first that made the mark), without using at detection the original or a copy of it, are not known until now.

Such a situation can be solved if the watermark is author and host dependent. In such a case the author will use at insertion and detection two keys: k_1 - author dependent and k_2 - signal/ host dependent. Using the keys he will generate a pseudo-random sequence k . The key k_2 , signal dependent, can be generated using *one-way hash* (OWH) functions. Such generators are including: RSA, MD4, SHA, Rabin, Blum/Blum/Shub [64]. The watermark extraction at the receiver is impossible without knowing the keys k_1 and k_2 . The k_2 key, being host dependent, the counterfeiting is extremely difficult. In copyright protection, the pirate will be able to give to a judge only his secret key k_1 and not k_2 . The last key is computed automatically using the original signal by the insertion algorithm. The hash function being noninvertible the pirate will not be able to produce a counterfeit identical with the original.

4.8.4 Basic Principles of Watermarking

As shown in the Introduction, watermarking has two basic processing: one at the sender and the other at the receiver:

- *Watermark insertion* in the host data. The insertion is done respecting the perceptual transparency and robustness requirements.

- *Watermark extraction (detection)* from the marked received signals (possibly altered) *and the comparison* between the extracted watermark and the original one, in case of deadlock.

For the robustness demand the watermark will be inserted using one or several robust cryptographic keys (secret or public). The keys will be further used at watermark detection.

The perceptual transparency is done according to a perceptible criterion, the last one being implicit or explicit. Therefore the individual samples of the host signal (audio signals, pixels or transform coefficients) used for the insertion of the watermark information will be changed only between some limits situated below the perceptiveness thresholds of the human senses (seeing, hearing).

Transparent insertion of the watermark in the digital host signal is possible only because the final user is a human being. His senses (hearing, seeing) are imperfect detectors characterized by certain minimal perceptiveness thresholds and by the *masquerade* phenomenon. By masquerade, a component of a given signal may become imperceptible in the presence of another signal called *masquerading signal*. Most of the coding techniques for audio and video signals are using directly or indirectly the characteristics of the *HAS - human audio system* or *HVS - human visual system* [72].

The watermarking techniques cannot, therefore, be used for data representing software or numbers, perceived by a computer (machines, not humans).

According to the robustness demand the watermarking signal (despite the small amplitude required by the perceptual transparency demand) is spread over several samples according to the granularity demands. This makes possible the detection of the watermark signal although the data is noise affected.

Fig. 4.25 and Fig. 4.26 are showing the bloc schemes of watermark insertion and detection.

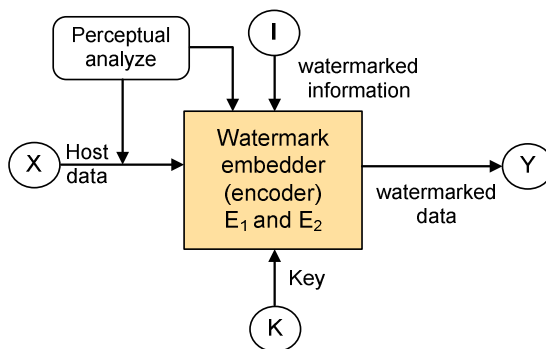


Fig. 4.25 Bloc scheme for watermark insertion

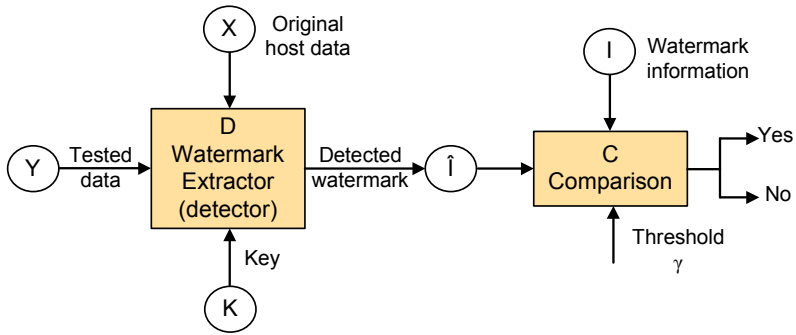


Fig. 4.26 Bloc scheme for watermark extraction and comparison

Watermarking (Fig. 4.25) consists in:

- Watermark information (I) generation (payload)
- Watermark generation (distinct from I – watermark information): W, that will be inserted into the host signal X; usually W depends on the watermark information (I) and on the key K:

$$W = E_1(I, K), \quad (4.61)$$

where E_1 is a function (in most cases modulation and spreading).

There are applications where, in order to limit the IBM attack, the watermark signal can be host signal X dependent):

$$W = E_1(I, X, K) \quad (4.62)$$

- *Key generation*; the key can be public or secret, leading to a possible classification of the watermarking techniques in public keys systems and private keys systems.
- *Watermark signal (W) insertion* in the host signal (X). The insertion is made with respect to the robustness and perceptual transparency demands, giving the watermarked signal Y:

$$Y = E_2(X, W), \quad (4.63)$$

where E_2 is a function (which usually makes a modulo 2 summation between W and X).

As a conclusion, in order to fulfil the perceptual transparency demands, the two models HAS or HVS, are taken into account directly or indirectly for watermarking. For the robustness requirements, the watermark information I is spread over the host data (see the granularity concept).

Watermarking can be done in the transform domain or in the spatial domain. It follows that, before watermark insertion or extraction, the host data needs to be converted in the domain where the processing will take place: spatial, wavelet, DCT (Discrete Cosine Transform), DFT (Discrete Fourier Transform), fractals. Each domain has specific properties that can be used in the watermarking process [47].

Watermarking can also be done for compressed or uncompressed host data; most applications are, however, for uncompressed data [34].

Remark

Due to the perceptual transparency demands, the changes in the host data are relatively small, so the watermark signal W , will be error vulnerable. In order to overcome this drawback, in transmission or storage, several protection measures can be taken using error correcting codes before watermark insertion, [2] and [32].

Watermark extraction (Fig. 4.26)

The watermark detector input signal is Y' and it can be the result of a watermarked signal with errors or not. In order to extract the watermark information \hat{I} , the original signal X is necessary – or not - depending on the detection type: (Fig. 4.26):

$$\hat{I} = D(X, Y', K) \text{ - nonoblivious detection} \quad (4.64)$$

$$\hat{I} = D(Y', K) \text{ - oblivious detection} \quad (4.65)$$

In copyright applications, the detected watermark information \hat{I} is compared with the ownership original I :

$$C(I, \hat{I}) = \begin{cases} \text{yes, if } c \geq \gamma \\ \text{no, if } c < \gamma \end{cases} \quad (4.66)$$

In practice, the comparison is made by a correlator that computes the cross-correlation c between I and \hat{I} , and a threshold detector with γ threshold value.

Most *watermarking techniques* are based on the principle of *spread-spectrum* [80], [32].

The idea of a spread spectrum system is to transmit a low bandwidth signal (in this case the watermark information I) on a large bandwidth channel with interferences (the audio or the video signal X).

The major advantages of a spread spectrum transmission are:

- the interferences are reduced
- the signal is hidden against interceptions.

- Watermark insertion:

The watermark information I is spread using a modulation with a pseudo-random noise, the watermarked regions being hidden in this way. In order to avoid processing that can eliminate the watermark information the insertion regions must be known (or at least recognizable). The spread spectrum techniques ensure an efficient protection of the watermark especially against usual, non-intentional data manipulation in transmission or storage (compression, scaling etc).

- Watermark extraction:

In spread spectrum based watermarking, the authorized detection (K is known) is easy to implement, even in the absence of the original X , using a *correlator*

receiver. The lack of synchronization can be eliminated using a *sliding correlator*. The maximum value of the correlation functions – the true value of the watermark information I - is found in this case by sliding.

We will illustrate the spread spectrum based watermarking principle for video signal in the spatial domain [32].

A video sequence can be represented as three-dimensional signal. Such a signal has two dimensions in the x-y axis and the third one is the time. For line scanning systems the signal can be viewed as a signal with a single dimension.

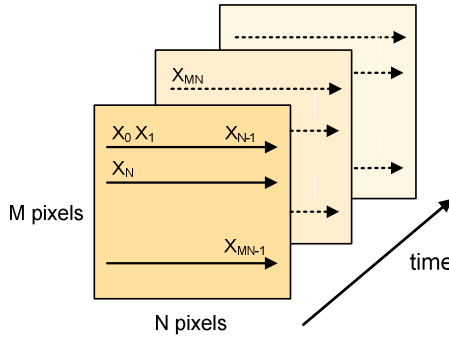


Fig. 4.27 Line - scanning video signal

- *Video signal watermark insertion (Fig.4.28)*

Let I be the watermark information with N_1 bits a_j :

$$I = \{a_j\}, j = \overline{1, N_1}, a_j \in \{-1, 1\}. \tag{4.67}$$

N_1 is the watermark payload and as we have seen its maximal value is $60 \div 70$ bits.

In order to make the watermark more robust this sequence I is spread over a large number of bits according to the chip rate (or spreading factor) m . Typical values for m are between $(10^3 \div 10^6)$ [81]. The spread sequence is then:

$$b_i = a_j, jm \leq i < (m+1)j, i \in N \tag{4.68}$$

The spread spectrum sequence, amplified with a local factor $\lambda_i \geq 0$, modulates a pseudo-random signal, the key K with p_i bits, $K = \{p_i\}$:

$$p_i \in \{-1, 1\}, i \in N \tag{4.69}$$

The watermarked spread sequence $W = \{w_i\}$

$$w_i = \lambda_i b_i p_i, i \in N \tag{4.70}$$

is summed with the video signal x_i . The video watermarked sequence is obtained:

$$Y = \{y_i\}, \quad i \in N \tag{4.71}$$

$$y_i = x_i + \lambda_i b_i p_i, \quad i \in N \tag{4.72}$$

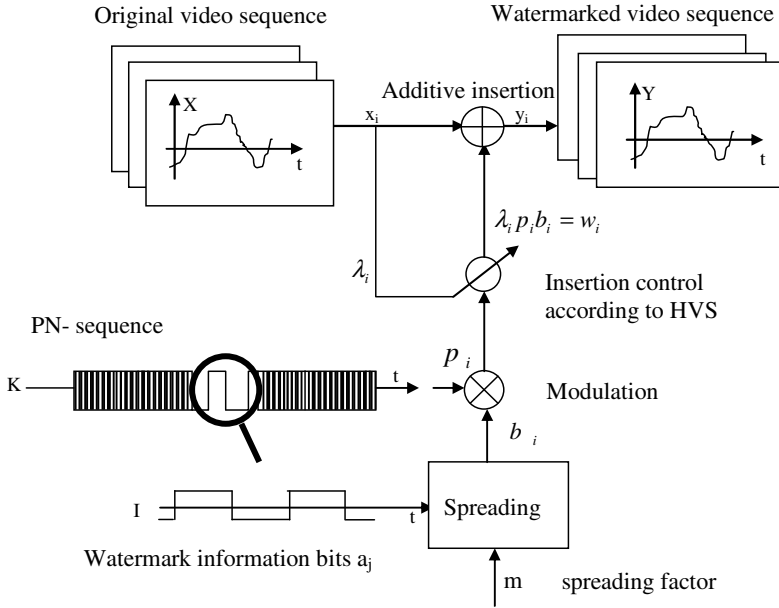


Fig. 4.28 Video sequence watermark insertion model

The pseudo-random sequence $K = \{p_i\}$ and the corresponding spread watermarked sequence $W = \{w_i\}$ are pseudo-random signals being therefore difficult to detect, localize and eliminate.

The security of a watermarking system depends on the key K . The key is chosen in order to insure the cryptographic security. Besides this, for correlative detection, the key must have good correlation and orthogonality properties. In such a way several watermarks can be embedded in the same host signal and these distinct watermarks can be detected without ambiguity at the receiver [80], [81].

The amplification factor λ_i , depends on the local properties of the video signal. In order to make the amplitude of the local watermark amplitude maximal, with respect to the perceptual transparency demands, the amplification factor λ_i can use HV spatial and temporal masquerade phenomena.

The watermark information is inserted in the least visible regions, for example in regions with fine details or contours.

- Video signal watermark extraction (Fig. 4.29)

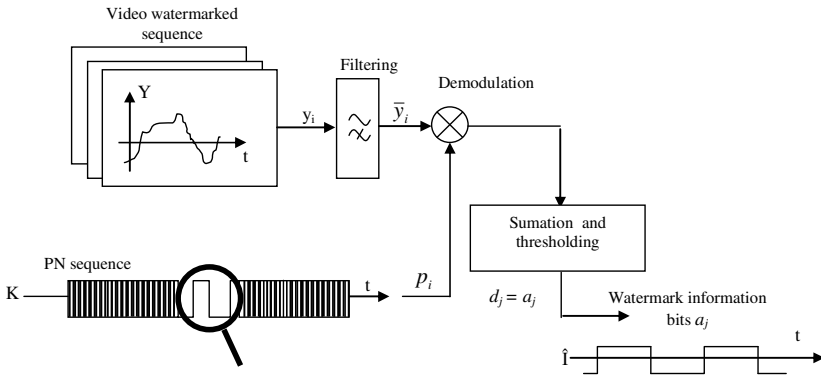


Fig. 4.29 Video sequence watermark extraction model

When the spread spectrum principle is used, the authorized extraction (the key K is known) can be easily done without knowing the original X , using a correlator.

Before demodulation (correlator based), the watermarked video sequence Y' (identical or different from the original one due to attacks or transmission processing) is high-pass filtered. The high pass filtering eliminates the major components from the video signal. High pass filtering is not compulsory but enhances the performances of the whole system, reducing the interferences between the video signal and the watermark signal.

Watermark detection consists in a multiplication between the watermarked video signal (Y') with the same key (K) used in the insertion process, followed by a correlation sum computation, d_j for each inserted bit and by a thresholding operation. An estimate (\hat{a}_j) for the watermark information bit a_j is thus obtained.

$$d_j = \sum_{i=jm}^{(j+1)m-1} p_i y_i = \sum_{i=jm}^{(j+1)m-1} p_i \overline{(x_i + \lambda_i b_i p_i)} \tag{4.73}$$

where \bar{y}_i is the value of the i -th high pass filtered video bit, supposed not affected by errors. Consequently:

$$y'_i = y_i = x_i + w_i \tag{4.74}$$

$$\underbrace{\sum_{i=jm}^{(j+1)m-1} p_i x_i}_{S_1} + \underbrace{\sum_{i=jm}^{(j+1)m-1} p_i \lambda_i b_i p_i}_{S_2} = S_1 + S_2 \tag{4.75}$$

S_1, S_2 are describing the contribution in the correlation sum of the filtered video signal and the filtered watermarked signal.

Considering that by high pass filtering the video signal x_i has been practically eliminated from y_i , the assumption $S_1 = 0$ is correct. Considering also that the influence of the high pass filtering on the watermarking signal is negligible, we have:

$$\overline{\lambda_i b_i p_i} \cong \lambda_i b_i p_i \quad (4.76)$$

Under these assumptions:

$$d_j \cong \sum_{i=jm}^{(j+1)m-1} p_i^2 \lambda_i b_i = \sigma_p^2 a_j m E[\lambda_i], \quad (4.77)$$

where σ_p^2 is the variance of the pseudo-random noise (K), and $E[\lambda_i]$ is the m pixels based average value of the coefficients λ_i .

The sign of the correlation sum gives the value of the watermark bit a_j in hard decision schemes:

$$\text{sign } d_j = \text{sign} \left(a_j \underbrace{\sigma_p^2 m \overline{\lambda_i}}_{>0} \right) = \text{sign} (a_j) \quad (4.78)$$

As seen from (4.78) the transmitted information bit is +1 if the correlation between the video signal containing the watermark bit and the pseudo-random sequence is positive, and -1 if the correlation is negative.

If the pseudo-random sequence used for the watermark extraction is different from the one used for watermark insertion (or if we do not have a synchronism between the two sequences) such a scheme will not work and the computed bits are random.

The lack of synchronization between the two sequences can be eliminated using a *sliding correlator*: every single possible sliding is done experimentally, the true value for a_j corresponds to the maximum value of the correlation s_j , obtained for a specific sliding.

Remark

The scheme given in Fig. 4.29 is an oblivious one, not needing the presence of the original X for watermark extraction. For a nonoblivious scheme, the use of the original, subtracted from Y before the demodulation – instead of filtering- will eliminate any interference between the video signal Y and the watermark W. The watermarking extraction in this case is much more robust.

The performances of this scheme can be estimated by computing the bit error rate BER (*Bit Error Rate*). One bit is in error if:

$$\text{sign} (S_1 + S_2) \neq \text{sign} (S_2) \quad (4.79)$$

This situation corresponds to:

$$\text{sign } S_1 \neq \text{sign } S_2 \text{ and } |S_1| > |S_2| \quad (4.80)$$

In [32] it is shown that the bit error rate is:

$$\text{BER} = \frac{1}{2} \operatorname{erfc} \left(\frac{\sigma_p \sqrt{m} E[\lambda_i]}{\sqrt{2} \sqrt{\sigma_x^2 + \mu_x^2}} \right), \quad (4.81)$$

where σ_x^2 and μ_x are the variance and the average value of the video signal, respectively.

From (4.81) we can see that the BER is small if m , σ_p and $E[\lambda_i]$ have high values.

In [32] - Table. 2. gives several examples of BER computed and measured for different values of m , and λ_i , with/without filtering.

The watermark information bit rate is:

$$R_{\text{WM}} = \frac{\text{number of luminance pixels per second}}{\text{spreading factor}} \quad (4.82)$$

Remarks

In order to maintain a certain value for the BER, even when attacks occur, the argument of the BER has to be raised using an insurance factor [32].

If m increases, R_{WM} decreases. If the number of the information bits per second is lowered correspondingly the watermark still rests robust to intentional or non-intentional attacks. The robustness can be explained by the fact that each information bit a_j is spread over a larger number of bits b_i .

If the spreading factor m decreases, the BER also decreases. In order to keep the BER between some admissible limits, even if m is small and R_{WM} relatively big, soft decoding error correcting codes can be used [72].

The method shown above for spatial domain video signals watermarking, can be used in any transformed domain of the video signal. Every transformed domain has its own advantages and drawbacks. A very good presentation of the state of the art in watermarking is made in [47].

4.8.5 Specific Attacks

4.8.5.1 Attack Definition and Classification

The causes leading to errors in the watermark extraction process are called *attacks*.

According to the way they were produced, the attacks can be classified in two major categories:

- *Unintentional attacks*, due to the usual signal processing in transmission or storage: linear (nonlinear) filtering, JPEG compression, MPEG-2 compression, pixel quantisation, analog to digital conversions, digital to analog conversions for recording processes, γ correction. A detailed description of these attacks is done in [26].

- *Intentional attacks* intentionally made in order to eliminate the watermark or to insert false watermark, keeping also the perceptual fidelity.

There is other attacks classifications among them we refer to: [81], [34]:

A. *Simple attacks*, the watermarked signal sustains some distortions, however the intention being not to eliminate the watermark. The majority of these attacks are unintentional attacks described above.

B. *Detection disabling attacks*, including the synchronization attack. These attacks are oriented towards the watermark extraction devices; the purpose of such an attack is to avoid watermark detection. A common characteristic for such attacks is the signal decorrelation, making the correlation based watermark extraction impossible. In this case the most important distortions are *geometric distortions*: zooming, frame rotation, sub-sampling, the insertion or extraction of a pixel or a group of pixels, pixel interchanges, spatial or temporal shifts.

In the case of the Stir Mark [69], the jitter attack consists in the elimination of some columns and the multiplication of others, keeping unchanged the image dimensions.

On the same category, frame modifications are included: frame removal, frame insertion or swapping.

C. *Ambiguity attacks*, also known as confusion, deadlock/ inversion-IBM/ fake watermark/ fake original attacks. These attacks are trying to create some confusion by producing a fake original.

D. *Removal attacks* are trying to decompose the watermarked image Y in a watermark W and an original X , in order to eliminate the watermark. In this category we mention *the collusion attack*, noise extraction and nonlinear filtering.

In multimedia MPEG compression based applications the attacks can be done in the compressed domain (frequency - DCT), or in the spatial domain. The most important attacks are done in the spatial domain, for uncompressed signals.

There are computer programs for several kinds of attacks, among them we mention:

- Stir - Mark, from Cambridge University,
- Attack, from University of Essex,

still images oriented useful also for dynamic images too.

In the following paragraphs we will expose the principle for two of the most powerful attacks: the inversion (IBM) attack, and the collusion attack.

4.8.5.2 The Inversion Attack / IBM / Confusion / Deadlock / Fake Watermark / Fake Original

The goal of this attack is the insertion, by faking, of some watermarks in an already watermarked signal, followed by a copyright claim of the fake owners.

The resulted signal contains several watermarks creating therefore a confusion related to the right owner (the first person who watermarked the document).

From a mathematics point of view this situation can be modeled as follows:
Let X be an original document watermarked by its owner (p):

$$Y_p = X + W_p \quad (4.83)$$

and Y_f the fake document, obtained by the forger by adding its own watermark W_f to the original already watermarked:

$$Y_f = Y_p + W_f = X + W_p + W_f \quad (4.84)$$

By proceeding on this manner, a new document with two watermarks is created, for the new document both the right owner and the forger could claim the copyright, each one having its own watermark inserted.

The question is, who is the right owner of the document X ?

A simple solution consists in using *the arbitrated protocol*. Let D_p and D_f be the watermark extraction algorithms used by the owner and the forger.

The judge verifies Y_p and Y_f using at the time the two extraction algorithms:

$$\begin{cases} D_p(Y_p) = D_p(X + W_p) = W_p \\ D_p(Y_f) = D_p(X + W_p + W_f) = W_f \end{cases} \quad (4.85)$$

and:

$$\begin{cases} D_f(Y_p) = D_f(X + W_p) = 0 \\ D_f(Y_f) = D_f(X + W_p + W_f) = W_f \end{cases} \quad (4.86)$$

The forger is identified, and thus, the problem posed in [72], is solved.

The above-mentioned problem becomes more difficult when the forger is able to produce a false (counterfeit) original: X_f - this case is known as the *inversion attack* or *IBM attack*.

Starting from a watermarked original:

$$Y_p = X + W_p, \quad (4.87)$$

a false original X_f is created by the forger, by extracting a false watermark W_f :

$$X_f = Y_p - W_f \quad (4.88)$$

It follows that:

$$Y_p = X_f + W_f, \quad (4.89)$$

Consequently the already marked image contains the forger's watermark; the forger can pretend now that Y_p is his, being created from the counterfeit original X_f and his watermark.

The above-described arbitrated protocol can no longer tell who is the right owner:

$$\begin{cases} D_p(Y_p) = D_p(X + W_p) = W_p \\ D_p(Y_f) = D_p(X + W_p + W_f) = W_f \end{cases}, \quad (4.90)$$

because Y_p is the same and it contains both watermarks.

This attack cannot be avoided in oblivious schemes (needing the presence of the original for extraction).

A necessary condition - but not sufficient - for avoiding the inversion attack is the use of *nonoblivious watermark extraction schemes*.

Basically there are two major nonoblivious extractions schemes:

1. the watermark has to be original dependent, using for example hash functions
2. the original signal X has to be time stamped at creation, the time stamp will permit to establish who is the right owner according to the temporal stamp's date.

1. *Oblivious watermark extraction schemes* using hash functions.

An original dependent watermark can be produced using one way hash functions, so that:

$$H(X)=h, \quad (4.91)$$

h represents the value of the hash function (hash value). Some robust hash functions based algorithms are MD4, MD5 and SHA [64]. It imposes the demand that *any watermarking system has to be oblivious*; it is evident that such a demand has to be also legally established in order to counterattack the IBM attack. In this case the forger can no longer generate a false original X_f because the watermark W_f cannot be dependent on the false original X_f .

2. Time stamping means that the watermark information has to contain some time stamp with information related to the certified original X , the certification date T , and also the name of the owner P , so such a time stamp is:

$$S = (h, T, P), \quad (4.92)$$

where h is the hash value of the function $H(X)$. Consequently the watermarked image is:

$$Y = X + W(S) \quad (4.93)$$

In this case the IBM attack is impossible, due to the fact that it is impossible to obtain:

$$X_f = Y - W(S_f) \quad (4.94)$$

4.8.5.3 The Collusion Attack

The goal of this attack is to eliminate the watermark, therefore to obtain an unmarked signal X .

Conditions: the existence of at least two copies of the same original differently watermarked:

$$\begin{aligned} Y_1 &= X + W_1 \\ Y_2 &= X + W_2 \end{aligned} \quad (4.95)$$

where W_1, W_2 are two digital fingerprints.

Actions

1. *Collusion by subtraction.* The difference between the two copies is done, so the watermarks are localized and therefore eliminated:

$$Y_1 - Y_2 = W_1 - W_2 \quad (4.96)$$

2. *Collusion by addition.* This is the general case for this type of attack; in this case the collusion is viewed as a mean operation performed on the same original marked with several different watermarks:

$$\begin{aligned} Y_1 &= X + W_1 \\ Y_2 &= X + W_2 \end{aligned} \quad (4.97)$$

The mean of the copies is:

$$\frac{Y_1 + Y_2}{2} = X + \frac{W_1 + W_2}{2} \quad (4.98)$$

From the above relation it can be seen that the watermark is half reduced; for a larger number of watermarked versions of the same signal, the original X can be obtained.

4.8.6 Applications

The main applications of the watermarking were mentioned in introduction. As shown in 4.8.3, each application has specific demands concerning the watermarking requirements. However, three important features define watermarking from other solutions:

- watermarks are perceptually transparent (imperceptible),
- watermarks are inseparable from the host data in which they are inserted, meaning that support the same processing as the host data.

The quality of the watermarking system is mainly evaluated by its:

- robustness, which indicates the capability of the watermark to “survive” to unintentional or intentional attacks,
- fidelity, describing how transparent (imperceptible) a watermark is.

The value of these quality parameters is highly dependent on the application. We will proceed presenting the most important applications, their requirements, the limitations of alternate techniques, emphasizing the necessity and advantage of the watermarking solution [27].

4.8.6.1 Broadcast Monitoring

- Who is interested in it?
 - advertisers, which are paying to broadcasters for the airtime; in 1997 a huge scandal, broke out in the Japanese television advertising (some stations usually had been overbooking air time),

- performers, to receive the royalties due them by advertising firms; in 1999 the Screen Actor Guild, after a check, found that the unpaid royalties in US television programs is aprox. 1000 USD/hour,
 - owners of copyrighted works, who want to be sure that their property is not illegally rebroadcast (major events, catastrophes, exclusive interviews, etc).
- Broadcast monitoring ways:
 - *using human observers* to watch the broadcast and record what they see and hear; these techniques is expansive, low tech and predisposed to errors.
 - *automated monitoring*:
 - *passive monitoring*: the humans are replaced by a computer that monitors broadcasts and compares the received data with those found in a data base with known works; in case of matching the work is identified. The advantages of this system are: do not require cooperation between advertisers and broadcasters and is the least intrusive. Unfortunately it has major disadvantages related to the implementation (need of huge data base, impractical to search); if signatures of the works are used, the data base diminishes a lot, but the broadcast altering the work, will make the signature of the received work different as those corresponding to the original (an exact matching is impossible). This system is applied for marketing, but not for verification services, because its error rate.
 - *active monitoring*: it removes the main disadvantages of the passive monitoring. There are two ways to accomplish this aim: using computer recognizable *identifiers* (e.g. Vertical Blanking Interval for analog TV or file headers for digital format) along with the work or the *watermarking*. The first solution has as main disadvantage the miss of guarantee in transmission and the lack of resistance to format changes. The watermarking solution, even more complicated to be embedded (compared to file headers or VBI), has not any more the risk to be lost as identifiers are.

4.8.6.2 Owner Identification: Proof of Ownership

Textual Copyright notices, placed in visible places of the work, are the most used solutions for owner identification. An exemplification for visual works is:

“Copyright data owner”

“© data owner” (© 2002 by Academic Press) the textual copyright notice for “Copr. data owner” [27].

Unfortunately this widespread technique has some important limitations: the *copyright notice* is *easily omitted* when copied the document (e.g. photocopy of a book without the page containing the copyright notice), or cropped (Lena is a cropped version of November 1972 Playboy cover) and, if not removed, it is *un-aesthetic*. The watermarking is a superior solution, eliminating the mentioned limitations due its proprieties of inseparability from the work (ca not be easily removed) and imperceptibility (it is aesthetic).

A *practical implementation* of a watermarking system for this application is *Digimarc*, distributed with Adobe Photoshop image processing software; when *Digimarc*'s detector recognize a watermark, it contacts a central data base over the Internet and uses this information to find the image owner. This available practical system is useful only for honest people.

When malicious attacks are done, as collusion for removal and IBM for creating a fake owner, the *proof of ownership* is a major task. In a copyrighted work a forgery is very easy to be done in both cases presented before. As example:

"© 2000 Alice" can easily be transformed by an adversary (pirate, traitor) Bob as: "© 2000 Bob" by a simple replacement of owner's. Using the *Digimarc* system, the replacement of Alice watermark with Bob watermark is not difficult (if a watermark can be detected, probably can be removed to). For this situation there are some solutions as using an arbitrated protocol (which is costly) or using work dependent watermark, as presented in chapter 4.8.5.

4.8.6.3 Fingerprinting (Transaction Tracking)

The digital technology allows making quickly cheap and identical (same quality) copies of the original and to distribute them easily and widely. Distribution of illegal copies is a major problem of our days, implying always money (huge amount) loses of copyright owners.

Fingerprinting is a new application, aiming to catch the illegal distributors or at least making its probability greater. A legal distributor distributes to a number of users some copies, each one being uniquely marked with a mark named *fingerprint* (like a serial number). A number of legal users, called pirates, cooperate in creating illegal copies for distribution. If an illegal copy is finding, the fingerprint could be extracted and traced back to the legal owner of that copy.

The idea of transaction tracking is old, being used to unveil spies. To suspects were given different information, some of them false, and by the action of enemy those who revealed the information were caught.

A *practical implementation* of a fingerprinting system was done by *DIVX Corporation*, now defunct [27]. Each *DIVX* player placed a unique watermark (fingerprint) into every played video. If such a video was recorded and than copies sold on the black market, the *DIVX Corporation* could identify the pirate by decoding the fingerprint. Unfortunately no transaction tracking was done during the life of *DIVX Corporation*.

Transaction tracking in the *distribution of movie dailies* is another application. These dailies are very confidential, being not allowed to the press. Studios prefer to use fingerprinting, and not visible texts for marking, because the last one are very easily removed. The quality (fidelity) is not necessary to be very high in this application.

4.8.6.4 Content Authentication (Fragile Watermarking)

The aim of this application is to detect data tampering and to localize the modification.

Why is it so important?

Because in digital, tampering is very easy and very hard to detect (see image modification with Adobe Photoshop). Consequences of such tampering could be dramatically in a police investigation!

Message authentication [64] is a basically problem of cryptography, a classical solution being the digital signature, which is an encrypted summary of the message.

This solution was used in “*trustworthy digital camera*” [27] by computing a signature inside the camera. These signatures act as a header (identifier) necessary to be transmitted along with the work. Unfortunately the integrity of this header is not guaranteed if format changes occur. As in broadcast monitoring, instead of a header, which could be lost, an incorporated signature in the work is preferred, realized using watermarking. These embedded signatures are authentication marks. These marks become invalid at the smallest modification of the work and for this reason they are named *fragile watermarks*.

In this application, robustness is not anymore a requirement; the watermark designed for authentication should be fragile, as nominated. A detailed presentation of this application is done in [27], chapter 10.

4.8.6.5 Copy Control

If the former described applications had effects after an intentional forgery, the copy control is aimed to prevent people to make illegal copies of copyrighted works. The ways for this type of protection mentioned in the Introduction are encryption and watermarking. The both, without an appropriate legislation will not solve the problem.

In the next chapter the first trial for an appropriate standard proposal is presented.

4.8.7 The Millennium Watermark System [53]

4.8.7.1 Introduction

The first attempt of standardization for the DVD’s copy protection is the Millennium watermarking system introduced by Philips, Macrovision and Digimarc in USA; it was submitted to the approval of the USA Congress, and the result was the “Digital Millennium Copyright Act” signed by the president of the USA in 28. 10.1998.

The main *cause* was the market explosion of digital products like DVDs, digital broadcasting of multimedia products and the producers’ exposure to potential huge financial losses, in case of the non-authorized copying.

The standardization of the video DVD’s provoked unusual debates in copy protection (like the 1970 ÷ 1975 years for the DES standardization) influencing the whole multimedia world.

On a DVD the information is encrypted (with secret algorithms) but in order to assure the copy protection the encryption is not enough. Using encryption on a storage device: CD, DVD, or in transmission on communication channels or open

interfaces copy protection can be realized using an authentication system and a session key generating mechanism for all interfaces (end to end encryption). Encryption used on DVDs supposes that the players or recorders have incorporated compliant devices. When the content is displayed in clear on a monitor or played on speaker (to the human consumer) the encryption-based protection disappears. It is now when the need for watermarking becomes clear; the watermark assures that copying is allowed for a restricted number of copies (one copy) or prohibited (never copy).

The basic demands for the DVD copy protection watermark system are:

- invisible and hard to remove
- fast watermark extraction (maximum 10 s), therefore real time processing
- cheap watermark detector, with minimum additional hardware required for players and recorders
- robust detection, the watermark has to resist when usual processing of the signal are performed: compression, noise adding, shifts, format conversions etc.
- the watermark's payload has to be at least 8 bits/ detection interval
- the false alarm probability - watermark detection without watermark signal - has to be below 10^{-12} .

4.8.7.2 Basic Principle of the Millennium System

For any practical implementation solution of the system, the basic demands are:

- cheap and simple,
- robustness with perceptual transparency fulfilled.

According to these demands, from the wide spectrum of technical solutions, the following one has been chosen: the real time detection in the spatial domain using a simple spatial correlator. Even if the transformed domain (DCT or wavelet) offers several advantages, the needed processing cannot be done in real time.

In order to reduce the complexity, pure spatial watermarking is used, each watermark sequence W is repeatedly inserted in each frame of the video signal. By proceeding on this manner one can view the video signal as a time accumulated still image. Watermark detection can be described consequently:

$$d = \frac{1}{NT} \sum_i \left(\sum_t y_{t,i} \right) w_i, \quad (4.99)$$

where: t is the temporal position of a pixel

i is the spatial position of a pixel

N is the number of pixel from an image (frame),

T is the number of frames used in the detection process.

Using time accumulation, the number of multiplying is reduced, and consequently the complexity and also the processing time are also reduced.

Block schemes for the watermark insertion and watermark detection are given in Fig. 4.30.a and Fig. 4.30.b, respectively. Both schemes are using the principle illustrated in Fig. 4.28 and Fig. 4.29:

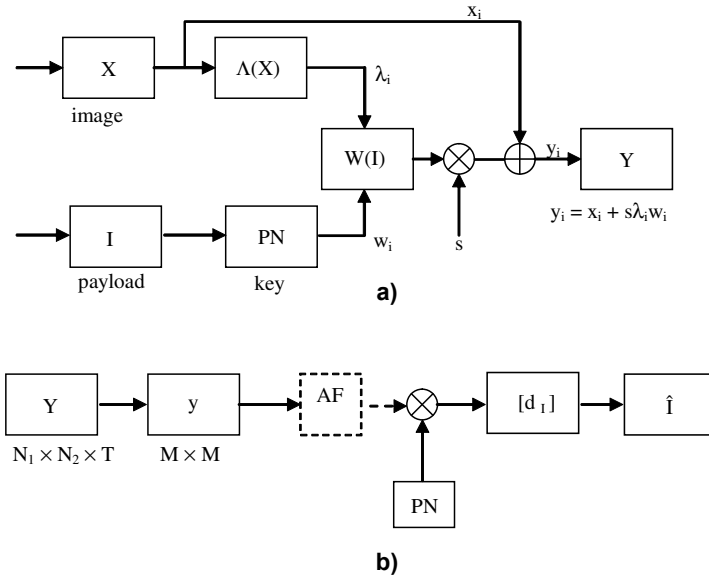


Fig. 4.30 Millennium Watermark block schemes: a) insertion, b) detection

$N_1 = 720, N_2 = 480, T = 27, M = 128.$

Watermark insertion (fig. 4.30.a)

The insertion of the same watermark in a given number of consecutive video frames is a 1 bit (mark) information for a video sequence.

A watermarked image $Y = \{y_i\}$ is obtained using:

$$y_i = x_i + s\lambda_i w_i, \tag{4.100}$$

s being a global scaling parameter, and $\Lambda = \{\lambda_i\}$ a local image dependent scaling factor. The values for λ_i are smaller in the “quiet” regions within the image and bigger in the “active” ones (contours, peaks). A satisfactory local scaling factor is obtained when the image is Laplace high pass filtered and taking the absolute value:

$$\Lambda = |L \otimes X| \tag{4.101}$$

the sign \otimes represents the cyclic convolution, and

$$L = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{4.102}$$

From (4.100) we can see that the (spread) watermark sequence W is a simple additive noise. Being formed using a pseudo-random sequence, its spectrum represents a white noise.

Watermark detection (fig. 4.30.b)

As shown above, the watermark detection is done using a spatial correlation, it follows that the watermarked image is:

$$Y = X + s \times \Lambda \times W \quad (4.103)$$

Then the correlation sum used for the detection is:

$$d = \frac{1}{N} \sum_i x_i w_i + \frac{1}{N} \sum_i s \lambda_i w_i^2 \quad (4.104)$$

Remark

In (4.104) it is assumed that during detection there is synchronization between the image and the watermark. For a correct aligning, the correct position is searched taking into consideration all the allowed spatial shifts, the decision variable d_k being:

$$d_k = \frac{1}{N} \sum_i y_i w_{i-k} \quad (4.105)$$

For a real time detection the search for all the possible shifts is computationally prohibitive. The solution used in Millennium is the insertion of translation symmetry in the structure of the watermark:

$$w_{i+k} = w_i, \quad (4.106)$$

for any vector k , which components are, multiplies of M , M being the dimension (tile) of the processing window support. The chosen practical value is $M = 128$.

Under these hypotheses the exhaustive search for all the possible shifts greatly simplifies. Due to the fact that the watermark is repeated over values that are multiplies of M , the watermarked image is superposed on an $M \times M$ matrix $B = \{b_i\}$. In this case

$$d_k = \frac{1}{M^2} \sum_i b_i w_{i-k} \quad (4.107)$$

Actually (4.107) represents a cyclic two-dimensional convolution.

$$D = B \otimes \underline{W}^* \quad (4.108)$$

(4.108) can be easily computed in the frequency domain [53]:

$$D = \text{IFFT} (\text{FFT}(B) \times \text{FFT}(\underline{W})^*) \quad (4.109)$$

In (4.109) FFT (Fast Fourier Transform) and IFFT (Inverse FFT) are the discrete Fourier transform and the inverse discrete Fourier transform, computed through a fast algorithm.

Remark

The detection performances can be improved if an adaptive filtering precedes the convolution.

From an experimental point of view the best detection is obtained by ignoring the amplitude information from \underline{W} and retaining only the phase information:

$$D = \text{IFFT}(\text{phase}(\text{FFT}(B)) \times \text{phase}(\text{FFT}(\underline{W}))^*) \quad (4.110)$$

Such a detection method is known in pattern recognition, as the symmetrical phase only filtering - SPOMF.

4.8.7.3 Millennium Standard Implementation

The real time watermark detector was built on three separate platforms:

- on a high-end Silicon Graphics workstation,
- on a Tri Media processor based board
- on a FPGA board.

The author shows in [53] that for DVD the best implementation is the FPGA based one. The costs associated with the implementation of the Millennium standard are given below:

	IC	ROM	RAM
FPGA	17kG	34kB	36kB
Silicon	14kG	1kB	36kB/3,6mm ²

Remarks

- the data from Table 1 are related to a single watermark detector
- in general the watermark detector has to be implemented on a MPEG decoder or a host-interface chip of a DVD-drive ; in both cases the functioning of the IC`s - (MPEG - decoding and PCI - bus streaming integrated circuits) needing huge memory buffers.

Even though there are no written information about the robustness of the Millennium system some conclusions tells that it fulfils the desired goal.

4.8.7.4 Unsolved Problems

The watermark detector can be placed in two places:

- From a security point of view the place of the copy control is in the driver of the DVD player, in the closest place to the storage medium (DVD). The PC data bus receives data from these DVD drivers, writing them into sectors without any interpretation possibility (audio, video or other).

For such a placement of the watermark detector, the data written into sector has to be recognized:

- data concatenation from different sectors
- decryption
- demultiplexing
- partial MPEG decompression.

The above-mentioned processing are nor usually done in the driver.

- The second solution is the MPEG decoder; in such a case the line between the driver and decoder has to be secure (encrypted). It seems that this solution is preferred.

4.8.7.5 Copy Control [53]

There are two basic principles:

- a) The *remarking concept*, consisting in the insertion of a second watermark by the recorder.
- b) The *ticket concept*, consisting in volatile information, lost in the copying process, like in the case of a bus ticket that loses its validity by obliteration.

The ``ticket`` acts like an authorized identifier. In order to assure that the ticket is specific to certain information, and to a certain transfer - for example copying - the ticket is encryptionally tied with the payload.

4.8.7.6 Media Type Recognition

According to the usage type, several media types can be used:

- pressed storage media (ROM), that can not be recorded
- recordable storage media.

The media type recognition can be regarded as a solution used in order to prevent ROM type discs recording.

4.8.8 Some Remarks

Digital watermarking was presented as a solution for copyright protection and especially for multimedia product unauthorized copying. In fact, even though several solutions were proposed, actually the domain rests untested, not experimented.

Among the great lacks shown, we can remind in the first place:

- the lack of standardization in algorithm testing and evaluation (lacks of benchmarking) (something like StirMark)
- the lack of a suitable legislation.

The copyright protection problem [72] shows that watermarking is by no means an unfailing method. Any research teams (even the whole technical community) will not solve copyright protection, because it is related to several legal aspects including a concise definition for similarity and subsequent work. Now we are in a period of great interdisciplinary efforts for national and international recommendations and standard elaboration for ownership protection in the digital era, in which both the multimedia products manufacturers and the legislation (the political factors) have to arrive to an agreement.

4.9 DNA Cryptography

4.9.1 Introduction

DNA cryptography and steganography is a new field, born from L. M. Adleman research [1] and Viviana Risca project on DNA steganography [74].

The huge advantages that DNA structure offers for efficient parallel computation and its enormous storage capabilities, made from this field a very promising one for various applications despite today limitations: expensive or time consuming.

For cryptographic purposes the interest is to generate very long one time pads (OTP) as cryptographic keys, which ensures the unbreakability of a crypto-system [66], to convert the classical cryptographic algorithms to DNA format and to find new algorithms for biomolecular computation.

Taking into account the huge advances in DNA technology, especially in microarray, the bio-processor [63], obeying an accelerated Moor's law [64], we must expect a faster repetition of microprocessor evolution and at larger scale.

4.9.2 Backgrounds of Biomolecular Technologies

The deoxyribonucleic acid (DNA), the major support part of genetic information of any organism in the biosphere, is composed by two long strands of nucleotides, each of them containing one of four bases (A – adenine, G - guanine, C – cytosine, T – thymine), a deoxyribose sugar and a phosphate group. The DNA strand leave chemical polarity, meaning that on each end of a molecule there are different groups (5' - top end and 3' - bottom end) [63].

A DNA molecule has double stranded structure obtained from two single-stranded DNA chains, bonded together by hydrogen bonds: A=T double bond and C≡G triple bonds. The double helix structure is configured by two single antiparallel strands (Fig. 3.10 paragraph 3.3.5).

The DNA strands can be chemically synthesized using a machine known as *DNA synthesiser*. The single stranded chains obtained artificially with the DNA synthesiser are called *oligonucleotides*, having usually 50-100 nucleotides in length. In what follows, the individual strands will be referenced as single

stranded DNA (ssDNA) and the double helix as double-stranded DNA (dsDNA). Individual ssDNA can, under certain conditions, form dsDNA with complementary ssDNA. This process is named *hybridization*, because the double stranded molecules are hybrids of strands coming from different sources (Fig. 4.31).



Fig. 4.31 Hybridization process

Ribonucleic acid (RNA) is a single strand of ribonucleotides, identical with ssDNA strands except thymine (T) which is replaced with uracil (U).

The genetic information flows from DNA into mRNA (messenger RNA) process known as *transcription* and from mRNA to protein, process called *translation*, defining what is known as the central dogma of molecular biology (Fig. 3.9 from paragraph 3.3.5) [63].

Gene is a segment of DNA that contains coding sequences (*exons*) and non-coding sequences (*introns*).

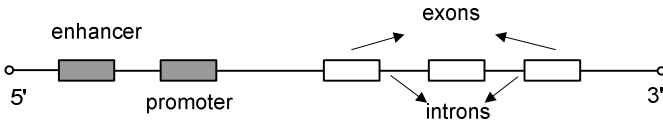


Fig. 4.32 Gene representation

When a gene is active, the coding sequence is copied in the transcription process in mRNA and, in the translation process, the mRNA directs the protein segments via genetic code (see 3.3.5). Transcription is governed by *regulatory elements* (enhancer, promoter), which are short (10 – 100 bp) DNA sequences that control *gene expression*.

Chromosome is a large organized structure of DNA coiled around proteins, containing genes, regulatory elements and other nucleotide sequences. It replicates autonomously in the cell and segregates during cell division (Fig. 4.33).

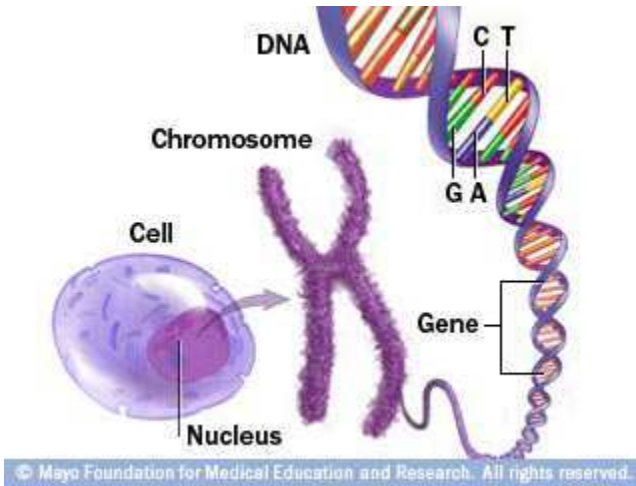


Fig. 4.33 Chromosome representation (<http://www.ohiohealth.com/>)

The entire DNA content of a cell, including nucleotides, genes and chromosomes, are known as the *genome*. Each organism contains a unique genomic sequence, with a unique structure.

Polymerase chain reaction (PCR) is a molecular biology technique used to exponentially amplify certain regions of DNA using enzymatic replication and starting with the DNA fragment (*primer*) to be amplified (Fig. 4.34).

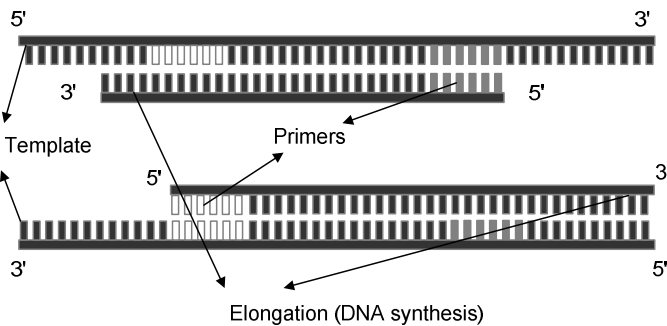


Fig. 4.34 Amplifying process in PCR technique

Recombinant DNA technology (gene splicing, genetic engineering) uses enzymes to cut and paste DNA “recombinant” molecules. Recombinant DNA enables the isolation and cloning of a gene and PCR its amplification (Fig. 4.35). dsDNA is cut with restriction enzymes (e. g. *E.coli*) that recognise specific nucleotides (*recognition sequence*) in it and cleave its double strand in precise location, leaving uncompleted “sticky ends” [63].

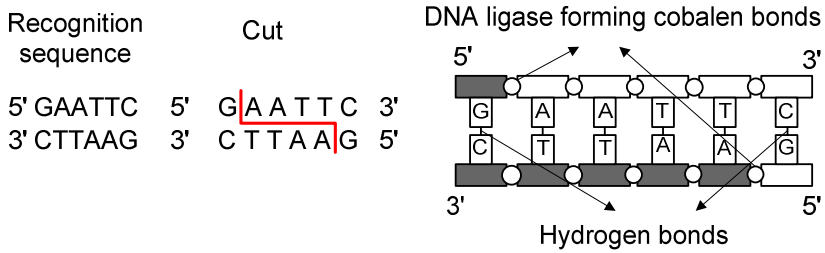


Fig. 4.35 Illustration of DNA recombinant process

Gel electrophoresis is a technique used to separate charged and labelled particles located in the gel (DNA, RNA etc) when electric current is applied. The DNA fragments of interest can be cut out of the gel and extracted or can be removed from the gel using Southern Blotting [4].

Microarray is the biological microprocessor. It is an ordered array of microscopic elements (spots) arranged in rows and columns on a planar substrate (glass or silicon). Each spot contains ssDNA molecules attached to the glass substrate (*target*). These molecules allow binding to them, by hybridization the fluorescent probe molecules. The *gene expression* information is measured by the fluorescent intensity of each spot (Fig. 4.36) [63].

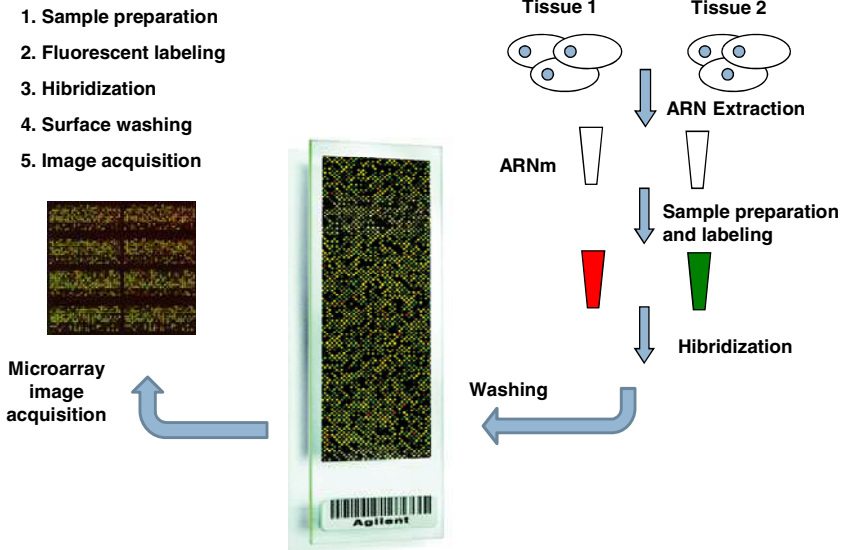


Fig. 4.36 Illustration of microarray experiment

This new concept presented in 1994, represents the collaboration of Mark Schena and Ron Davis (Stanford University and Affymetrix company). It is a revolutionary technique allowing an enormous increase in speed and precision of

gene quantitative analyses. A single microarray can be used to analyse the entire human genome (aprox. 25000 genes) in a single step of few minutes!

4.9.3 Elements of Biomolecular Computation (BMC)

BMC methods were proposed by Adleman [1] to solve difficult combinatorial search problems using the great available parallelism to the combinatorial search among a large number of possible solutions represented by DNA strands. In 1995 there were proposals to use BMC methods for breaking DES [9]. Besides the combinatorial search, BMC has many other exciting applications, due to the exceptional storage capacity of DNA. In cryptography the interest is for long OTP, for classic cryptographic algorithms in DNA format and also for new BMC algorithms.

4.9.3.1 DNA OTP Generation

A OTP in DNA format can be generated in two main ways:

- Assembling randomly long sequences from short oligonucleotide sequences. The ssDNA segments are bound together using a special protein (ligase) and a short complementary template (Fig. 4.37)

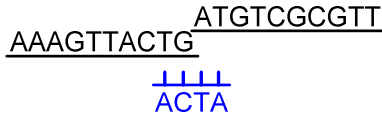


Fig. 4.37 Binding process between two ssDNA segments

- Using the chromosome sequence which is very large (thousands, millions bases), or segments of chromosomes delimitedated by primers (short length; 20 bp, DNA sequences) the distinct number of possible primers is 4^{20} , indicating the order of the brut force attack in this case.

4.9.3.2 Conversion of Binary Data to DNA Format and Vice Versa

In order to use the classic cryptographic algorithms in DNA format, conversion from binary to DNA format are required. DNA alphabet being of four letters (A, C, G, T), it is obvious that two bits are needed for encoding the four letters (Table. 4.9).

Table 4.9 DNA to binary conversion

DNA	Binary	ACSII - 7bits Decimal	ASCII - 8 bits Decimal
A	00	0	0+1 = 1
C	01	1	1+1 = 2
G	10	2	2+1 = 3
T	11	3	3+1 = 4

The plain text message is transformed in bits and after, in DNA format. If the message is a text, then the encryption unit is a character and it will be represented on ASCII code on 7 bits. If the message is an image than a pixel is the encryption unit and can be represented on 8 bits at least. Taking into account that each DNA letter (A, C, G, T) is represented with 2 bits, it means that an ASCII character or a pixel will be represented on 4 letters, being equivalent to a byte.

Example 4.11

Convert the message *secret* in ASCII, binary and DNA formats.

Solution:

Message	ASCII			DNA
	Decimal	Binary 7 bits	Binary 8 bits	
s	115	1110011	01 11 00 11	CTAT
e	101	1100101	01 10 01 01	CGCC
c	99	1100011	01 10 00 11	CGAT
r	114	1110010	01 11 00 10	CTAG
e	101	1100101	01 10 01 01	CGCC
t	116	1110100	01 11 01 00	CTCA

Remark

This example was given to understand the ASCII \rightarrow binary \rightarrow DNA conversion and it is not used for cryptographic purposes (it would be too easy to be broken).

4.9.3.3 DNA Tiles and XOR with DNA Tiles

DNA tiles [30] were synthetically designed after Wang tiles [82], using individual oligonucleotide chains which might hybridize in one helix, than cross-over and hybridize in another helix (Fig. 4.38).

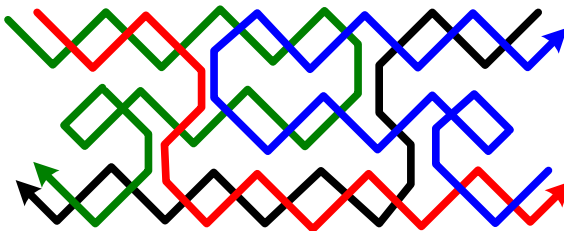


Fig. 4.38 Triple-helix tile

Upper and lower helices end with uncompleted sticky ends, used for binding other tiles with complementary sticky ends (Fig. 4.39).

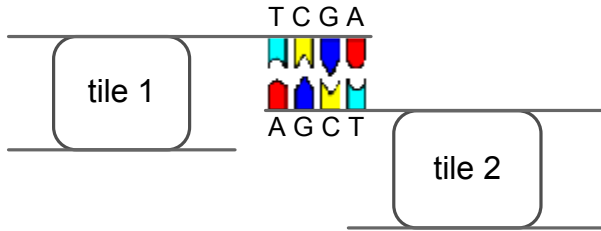


Fig. 4.39 Tiles assembling through complementary sticky ends

Binary data can be encoded using a single tile for each bit. The difference between 0 and 1 is given by the group of nucleotides on sticky ends.

Example 4.12

Tiles binding in a string will be discussed.

In order to make a string with DNA tiles, two steps are required:

- Selection of START tiles (Fig. 4.40)

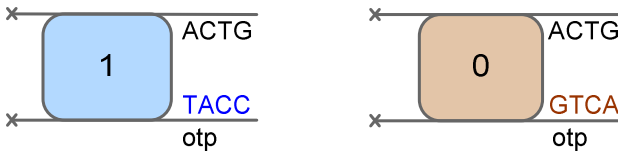


Fig. 4.40 Tiles for START bits in a string

- Selection of stacks with tiles for the rest of the bits in the string (Fig. 4.41).

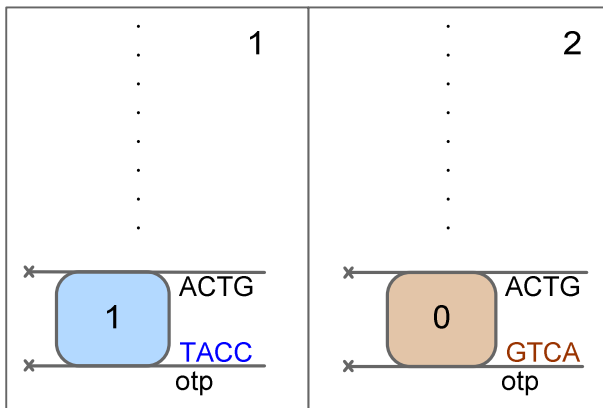


Fig. 4.41 Stacks with tiles for the rest of the bits in a string

- *DNA XOR with tiles*

In order to perform bit-wise XOR operation between two messages of given length (in cryptography it is XOR between the plain text and OTP key), a linear structure is obtained (Fig. 4.42)

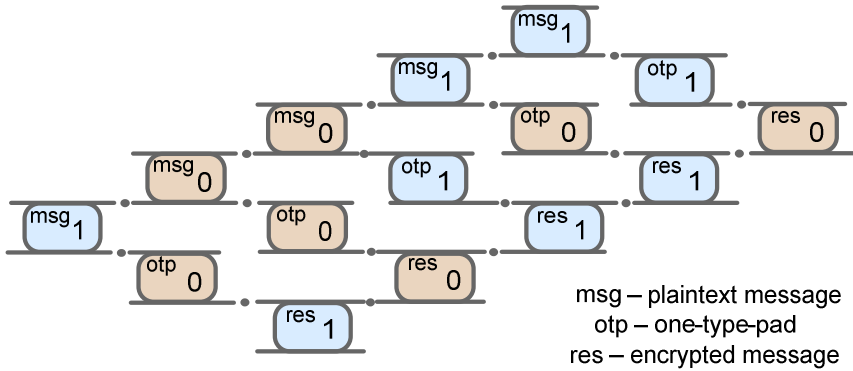


Fig. 4.42 XOR computation with tiles

The truth table of XOR function is given in Table 4.10.

Table 4.10 Truth table of XOR function

Inputs		Output
x_1	x_2	$y = x_1 \text{ XOR } x_2$
0	0	0
0	1	1
1	0	1
1	1	0

4.9.4 DNA Based Steganography and Cryptographic Algorithms

4.9.4.1 Steganography Technique Using DNA Hybridization

The technique of hiding a message in a DNA medium (microdot) was presented in 1999 in [74].

Based on sticker model for DNA computation [62] and on the technique of hiding a message in DNA microdots [74], a DNA encryption algorithm is presented [18]. A ssDNA OTP key is used to encode a text (plaintext). The technique to hide the encrypted message (ciphertext) could be the one presented in [74]. The steps of the algorithms are:

- a) *Conversion of the plain text in binary*
Message (TEXT) \rightarrow ASCII \rightarrow binary (n bits)
- b) *Generation of ssDNA OTP*
 - each bit will be encoded with 10 nucleotides, chosen randomly; the length of OTP will be greater than $10xn$.
- c) *Encryption*
 - for a binary “1” in the message, a strand of 10 bases long, complementary to the ssDNA OTP created at b) is made.
 - For a binary “0” in the message, neither operation is performed.
- d) *Hiding the ciphertext using DNA steganography techniques; the ciphertext is placed between two primers and hidden in microdot, for example, as in [74], using as background fragmented and denaturated DNA (fig 4.13).*

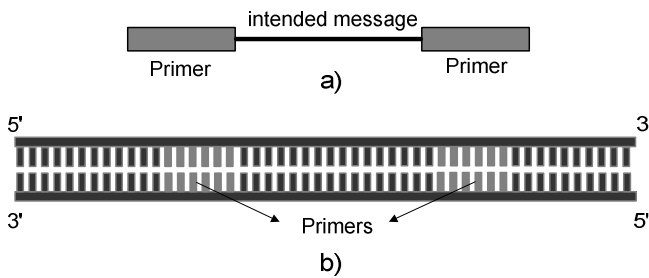


Fig. 4.43 Cipher text hiding: a) structure of cipher text inserted between two primers; b) dsDNA containing (hidden) the cipher text

- e) *Message recovery (decryption)*
The intended recipient requires the knowledge of the:
 - medium containing the hidden message
 - primers and OTP used for encryption
 Steps performed for the message recovery are:
 - PCR performing using right primers to amplify the hidden ciphertext
 - analyses of PCR product by gel electrophoresis or microarray analyses
 - cleavage of the obtained strand in segments of 10 bases long, using restriction enzymes, in order to cut the strand
 - hybridization between the obtained segments and the original OTP
 - reading the message: hybridized segments are “1” and ssDNA are “0”
 - OTP destruction.

Example 4.13

The previous algorithm will be exemplified for the message ZOO (except the steps requiring biomolecular technologies).

- a) Conversion from text \rightarrow binary
ZOO \rightarrow ASCII: 90 79 79 \rightarrow binary: 101101011001111110011111 – 21 bits = n

- b) Generation of ssDNA OTP: for each binary digit 10 nucleotides, randomly selected (eq. using function “*randseq (length)*” from MatLab Bioinformatics Toolbox). The sequence can be generated in DNA, RNA or amino alphabet. In our example sequence of 220 bases (> 210 bases) was generated:

TAAATATGTAACCCCCTGTAGCCAGCTTCCCTCCTCTACTGAGAG
 TCGGTAGTCGGCCTAACGTCACTCCTCAGCCTTGTTTCAGTAACAGTC
 GACCCCTAAGTGTCCCGATCGTCGGGAGTGTATGAGAGAGCAAAC
 TGTGATTAGCGCTAGCCCGAGTCCTTGCTTCTCACGCAATCAAGGAA
 TTGGTGTGTATATGCACTCGCGGGGTAATAGAGCA

- c) Encryption: for binary “1”s, complementary ssDNA are created and for binary “0”s neither operation is performed. In our case the encoded stream (corresponding to the 14 “1”s) is:

TGGGGGACAT, GAGGAGATGA, CTCTCAGCCA,
 TGCAGTGAGG, AGTCATTGTC, AGCAGCCCTC, ACATACTCTC,
 TCGTTTGAAC, ACTAATCGCG, ATCGGGCTCA, TCCTTAACCA,
 CACATATACG, TGAGCGCCCC, ATTATCTCGT

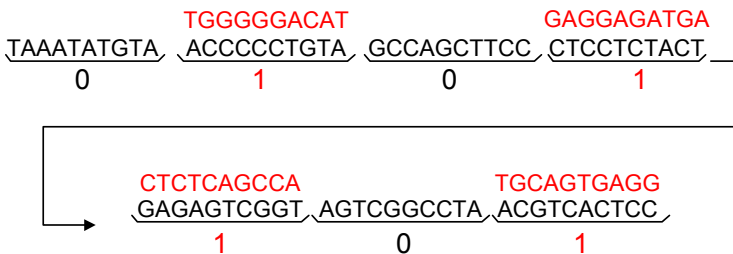
and the OTP containing the hybridized parts is:

```
TAAATATGTAACCCCCTGTAGCCAGCTTCCCTCCTCTACTGAGAGTCGGTAGTCG →
..... TGGGGGACAT ..... GAGGAGATGACTCTCAGCCA ..... →
→ GCCTAACGTCACTCCTCAGCCTTGTTTCAGTAACAGTCGACCCCTAAGTGTCCCGA
→ ..... TGCAGTGAGG ..... AGTCATTGTC .....

TCGTCGGGAGTGTATGAGAGAGCAAACCTGTGATTAGCGCTAGCCCGAGTCCTT →
AGCAGCCCTCACATACTCTCTCGTTTGAACACTAATCGCGATCGGGCTCA..... →

→ GCTTCTCACGCAATCAAGGAATTGGTGTGTATATGCACTCGCGGGGTAATAGAGCA
→ ..... TCCTTAACCACACATATACGTGAGCGCCCCATTATCTCGT
```

- d) Message recovery (decryption). The decryption of the first “101101” bits from the encrypted message ZOO is:



4.9.4.2 Chromosome DNA Indexing Algorithm

As shown in 4.9.3.1, OTP can be generated using chromosomes segments, situated between two primers, 20 bps long.

The steps of the algorithm are:

- a) *Generation of OTP* as session key for symmetric encryption, using a sequence of DNA chromosomes selected from an available public data base [38]
- b) *Encryption*
 - The plaintext, if text, is converted in ASCII code (on 8 bits: 0+7 bits), and, if image, in 8 bits at least; than the binary stream is converted in DNA format , as indicated in 4.6.3.2, meaning that each 8 bit character is transformed in 4 letters (A, C, G, T).

The OTP sequence is analyzed in steps of 4 bases long and compared to the plaintext in DNA format (Fig. 4.44).

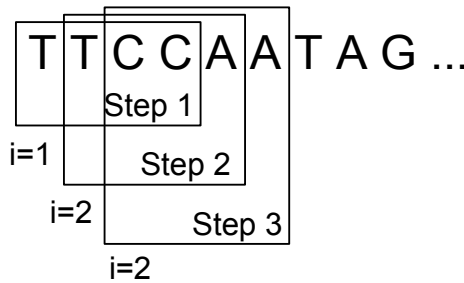


Fig. 4.44 Illustration of OTP scanning process for message encryption

- If 4-letters sequence, representing a character of the plaintext was retrieved in the chromosomal sequence, then the starting point (index in chromosome) of identical 4-letters is memorized in an array. For each 4-letters character an array of indexes in the chromosomal sequence is obtained. The number of indexes allocated to a character depends on the number of occurrences of the character in the chromosomal sequence. From this index array, a number is selected randomly and will be the encrypted character. Thus, the ciphertext is an array of random indexes.
- c) *Message recovery (decryption)*
 - The same OTP along with the primers used for encryption are required to decrypt the message. The steps at decryption, as for any symmetric algorithm are identical with those used for encryption, but made in reverse order.
 - This algorithm is using OTP key, meaning that a key is used only once. It is necessary to be able to generate great number of very long keys in order to leave a truly OTP. Chromosomes indexing offers a huge variety of possible genes and chromosomes itself can be selected from different organism, being thus excellent materials for cryptography, offering *randomness and non-repeating OTPs*.

- Transmission of the required OTP chromosome and primers, in fact the *key management*, which is the hardest problem in the symmetric cryptography, becomes *much easier*. Only the type of the chromosome and primers are needed to be known, which, being extremely small in length, could be transmitted using classic PKC for example.

The last two features look to be two great advantages of this algorithm. In fact this is not properly a DNA cryptographic algorithm because it is not using to DNA medium, but is using the huge randomness that DNA medium is offering. Serious studies to compare the classic algorithms towards this new one, will validate or not the efficiency for practical applications.

Example 4.14

Apply chromosome DNA indexing algorithm using Homo sapiens FOSMID clone ABC145190700J6 from chromosome X, available in NCBI database [38] for the message: *secret*.

Solution

- a) A fragment from sequence file in FASTA format is:

```
>gi|224967179|gb|AC234315.2| Homo sapiens FOSMID clone ABC14-50190700J6 from
chromosome X, complete sequence
TTCCCAATAGGCTGGACTGCTTACC&CCCATGTGGCCTCAAAGAGCTCCAGTCACTCCTTTACGA&CC
AATCACTCCAGA&ACTTTAGA&CAAAGTTTCTGAGTTACTCCTTGTAATAGGCTAAATAATGGCTCC&AAA
G&ATATTAGGATTTGATTCC&AGAACCTATA&AATATTACCTTATTTGG&AA&ACGGTTC&TAGCAGATG&TA
TTGAGTTA&AGGATATTGAGATG&CAGAGATTATTTTAGATTATCTAGACTATCTGGG&TGGATG&TATTGG&TC
AGGGTTCTTCAGAGGACAGAGCCAATAGGATATATGTATATA&AA&AGGGAGTTAATTAGG&GAG&AATTGGC
TCACATGATTAC&AGGTGA&AGTCCC&CGATAGG&CCCTCTG&CAA&ACTGGG&GAG&GAA&GCTAGTTG&TG&TGGC
TCAGTCC&AA&TCC&AA&AG&CCTCAA&AA&CTGG&G&A&AG&CTG&AC&AGTCA&A&AG&CCTAG&TCTG&AG&GCC&AA&AG&TC
CA&AG&AG&CC&CCTG&AG&AG&GCTG&CTG&GTG&CA&AG&TCC&AA&AG&TTC&AA&AG&GTTA&A&CA&AA&AC&CTG&A&AG&TCTG&GTG&TC
CAA&AG&GC&AG&G&AG&G&AG&G&A&AG&C&AG&AC&AG&G&A&A&G&A&A&G&CA&AA&C&AG&ACT&C&AG&CA&A&G&A&A&G&CTG&CTG&TTC
TTCC&AC&CTG&CTTTG&TTCT&AG&CC&AC&G&CTGG&C&AG&TCA&ATTG&CAT&GGT&G&CC&ATCC&AC&ACTG&AG&GGT&GG&ATCT
TC&CTC&TA&AC&AG&TCA&AA&C&ACTG&ACTCAA&ATG&TC&ATCTTCTG&G&CA&R&C&CC&CTC&AC&AG&AC&AC&CC&AG&AA&
CA&ATG&CTT&C&ACC&AG&CC&AT&CT&ATG&C&AG&CC&TCA&ATCC&AG&TCA&AG&GTG&AC&AC&CTA&AT&GG&TTA&AT&GG&TTA&TA&
ACC&AC&GG&TTA&ATA&AC&CAT&G&AC&AG&TGG&GTCT&AA&ATG&TA&ATC&AC&GTG&TATC&CTTATA&AA&AA&AA&AG&AG&GC&AG
AG&GG&AG&ATTTG&A&AG&AG&CTAT&AC&AG&AG&G&A&AG&CA&AC&GTG&A&AG&ATGG&AG&G&AG&G&A&A&ATTTGG&CC&AT&CA
```

b) Encryption

Be the plaintext: *secret* → ASCII: 115 101 99 114 101 116

s → 115 → 01110011 → CTAT → indexes:

166	258	789	927	1295	2954	3045
	3098	3181	3207	3361	3763	4436
	4559	5242	5443	5794	5938	5966
	7392	7698	7762	7789	7832	8128
	8627	9918	11871	12240	12332	12383
	12581	13107	13128	13324	14919	15169
	15177	15494	15602	15844	16073	16369
	16829	16891	16939	17227	17342	17718
	17818	18564	19530	20022	20437	20619
	21145	21411	21419	21725	22030	22051
	23157	23180	23231	23311	23367	23430

23434	23556	23811	24005	24038	24182
24568	25871	27176	27208	27896	29321
29642	29848	30087	30097	30110	30438
30472	31090	31487	33204	33226	33321
33378	33612	35520	35530	35646	35768

For each character was chosen a random index from its array of indexes. Below are established positions of random indexes inside character's arrays:

115 → 70th index 23811
 101 → 26th index 13981
 99 → 7th index 8011
 114 → 57th index 21195
 101 → 57th index 32741
 116 → 158th index 25264

Final encrypted message is:

23811 13981 8011 21195 32741 25264

c) Message decryption

The key is Homo sapiens FOSMID clone ABC14-50190700J6, from chromosome x complete sequence. First we read this sequence using functions from Bioinformatics Toolbox:

```
FASTADData = fastaread('homo_sapiensFosmid_clone.fasta')
```

Each index from received encrypted message was used to point in chromosomal sequence:

```
SeqNT=FASTADData.Sequence(i:i+3)
```

Using these pointers we extracted for each character a 4-bases DNA sequence. This variable was transformed in numerical value, using transformation offered by Matlab Bioinformatics Toolbox (A-1, C-2, G-3, T-4). As transformation starts with 1, at encryption to each digit was added a unit and after that it was transformed in base (example, "00" binary → 0 digit → 0+1 → A). At decryption from each obtained digit was subtracted a unit and after that transformed in 2 bits, for example:

CCCA (bases) → 2221(digits) → 2-1, 2-1, 2-1, 1-1 → 1110 (digits) → 01 01 01 00 (bits)

Obtained binary numbers are the ASCII cods of the recovered message characters.

4.9.4.3 DNA XOR OTP Using Tiles

Based on principle described on 4.9.3.3 and using the idea presented in [30] an encryption algorithm is presented in [78]. The steps of the algorithm are:

- a) Message tiles binding in a string.
- b) XOR operation between message (plaintext) and OTP (encryption key) (Fig. 4.45).

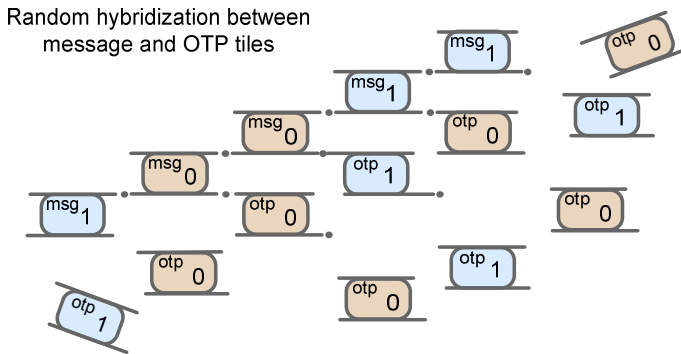


Fig. 4.45 Design of the One Time Pad tiles

c) Encrypted message results: the tiles as result of DNA XOR operation will have a pair of sticky ends for binding between them and a pair for binding to encryption key tiles (Fig. 8).

d) Cleavage process to keep the ciphertext only. In order to cut the ciphertext from the rest of tiles, we propose to use restrictive enzymes. Then the ciphertext will be stored in a compact form and sent to the destination together with the string of labels.

Since XOR is its own inverse; decryption is made using the same key and operations. The receiver needs an identical pad and use the received string of labels in order to extract from this pad the encryption key. After self assembling of the same tiles used for encryption, but in reverse order, the plaintext is extracted.

Example 4.15

DNA XOR OTP using tiles algorithm:

- a) Message tiles binding in a string
 - For the START tiles and the stacks for message binding the selection given in Example 4.2 is used.
- b) Microcontroller for tiles binding
 - acknowledgement of the message beginning
 - decision concerning the next bit from the message
 - verification of message ending

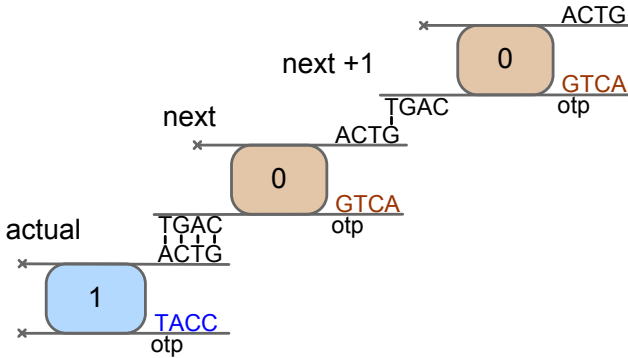


Fig. 4.46 Example of message tiles binding

c) XOR operation between message and OTP

OTP is a great set of labeled tiles representing 0 and 1. Labeling could be a certain order of nucleotides from the inside structure of the tile.

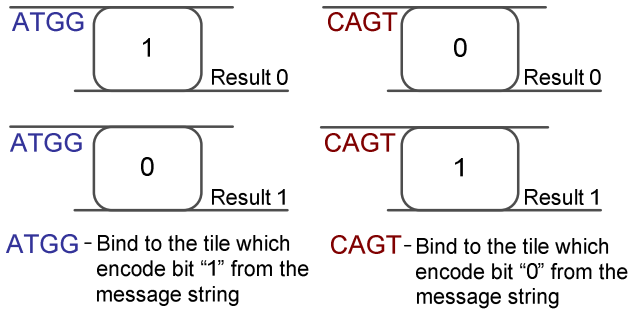


Fig. 4.47 Design of the one-time-pad tiles

References

- [1] Adleman, L.M.: Molecular computation of solution to combinatorial problems. Science 266, 1021–1024 (1994)
- [2] Ambroze, A., Wade, G., Serdean, C., Tomlinson, M., Stander, Y., Borda, M.: Turbo code protection of video watermark channel. IEE Proceedings Vision Image, Signal Processing 148(1), 54–58 (2001)
- [3] Angheloiu, I., Gyorfı, E., Patriciu, V.: Securitatea si protectia informatiei in sistemele electronice de calcul. Editura Militara, Bucuresti (1986)
- [4] Arizona Board of Regents and Center for Image Processing in Education, Gel Electrophoresis Notes What is it and how does it work (1999), <http://science.exeter.edu>
- [5] Bajenescu, T., Borda, M.: Securitatea in informatica si telecomunicatii. Editura Dacia (2001)
- [6] Bassia, P., Pitas, I.: Robust audio watermarking in the time domain. In: Proc. European Signal Proc. Conf., Rhodes (1998)

- [7] Berghel, H., German, L.O.: Protecting Ownership rights through digital watermarking. *IEEE Computer* 29(7), 101–103 (1996)
- [8] Bojcovics, Z.S., Toma, C.I., Gui, V., VasIU, R.: Advanced topics in digital image compression. Editura Politehnică, Timișoara (1997)
- [9] Boneh, D., Dumworth, C., Lipton, R.: Breaking DES using a molecular computer. In: Proceedings of DIMACS workshop on DNA computing (1995)
- [10] Boneh, D., Shaw, J.: Collusion-secure fingerprinting for digital data (963). In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 452–465. Springer, Heidelberg (1995)
- [11] Boney, L., Tewfik, A.H., Hamdy, K.H.: Digital watermarks for audio signals. In: Proc. Of 3rd IEEE ICMCS, pp. 473–480 (1996)
- [12] Borda, M.: Collaborative research agreement between the University of Plymouth UK and the Technical University of Cluj-Napoca Ro in Video Watermarking, nr. 33, November 11 (1999)
- [13] Borda, M.: Digital watermaking basics. *Acta Technica Napocensis* 42(1), 48–59 (2002)
- [14] Borda, M.: Principles of Watermarking. In: Proceedings of Enelko, Cluj-Napoca, pp. 31–40 (2003)
- [15] Borda, M., Major, R., Deac, V., Terebes, R.: Raport de faza la contract Contract ANSTI 6113/2001. Tema: Sistem de marcare a imaginilor numerice în scopul autentificării proprietății intelectuale asupra acestora (2001)
- [16] Borda, M., Major, R., Deac, V., Terebes, T.: Raport de faza la contract Contract ANSTI 6113/2002. Tema: Sistem de marcare a imaginilor numerice în scopul autentificării proprietății intelectuale asupra acestora (2002)
- [17] Borda, M., Tornea, O.: DNA Secret Writing Techniques. In: Proceedings of 8th Intl. Conf. on Communications, pp. 451–456 (2010)
- [18] Borda, M., Tornea, O., Hodoroagea, T., Vaida, M.: Encryption system with indexing DNA chromosomes cryptographic. In: Proceedings of IASTED, pp. 12–15 (2010)
- [19] Borda, M., Vancea, F., Deac, V.: Raport intern de cercetare. Contract ANSTI 6113/2000. Tema: Sistem de marcare a imaginilor numerice în scopul autentificării proprietății intelectuale asupra acestora (2000)
- [20] Brasil, J., Low, S., Maxemchuk, N., O’Gorman, L.: Electronic marking and identification techniques to discourage document copying. In: *IEEE Infocom 1994*, pp. 1278–1287 (1994)
- [21] Brasil, J., Low, S., Maxemchuk, N., O’Gorman, L.: Electronic marking and identification techniques to discourage document copying. *IEEE, J. Select. Areas Commun.* 13, 1495–1504 (1995)
- [22] Bultzer, P.L., Jansche, S.: Mellin Transform Theory and The Role of its Differential and Integral operators. In: Proceedings of Second International Workshop On Transform Methods and Special Functions, Varna, pp. 63–83 (1996)
- [23] Caroni, J.: Assuring ownership rights for digital images VIS 1995. Session Reliable IT Systems, Veiweg (1995)
- [24] Cox, I.J., Kilian, J., Leighton, T., Shamoon, T.: Secure spread spectrum watermarking for multimedia. Technical Report 95-10, NEC Research Institute, Princeton, NY, USA (1995)
- [25] Cox, I.J., Linnartz, J.: Public watermark and resistance to tampering. In: *IEEE Int. Conf. On Image Processing*, vol. 3, pp. 3–6 (1997)
- [26] Cox, I.J., Linnartz, J.: Some general methods for tampering with watermarks. In: *IEEE Int. Conf. On Image Processing*, vol. 16(4), pp. 587–593 (1997)

- [27] Cox, I.J., Miller, M., Bloom, J.: Digital watermarking. In: Principle and practice. Academic Press, London (2002)
- [28] Cox, I.J., Miller, M.L., Mckellips, A.L.: Watermarking as Communications with Side Information. *Proceedings of IEEE* 87(7), 1127–1141 (1999)
- [29] Craver, S., Memon, N., Yeo, B.L., Minerva, M.: Resolving Rightful Ownerships with Invisible Watermarking Techniques: Limitation, Attacks, and Implications. *IEEE Journal on Selected Areas in Communications* 16(4), 573–586 (1998)
- [30] Gehani, A., LaBean, T.H., Reif, J.H.: DNA-based cryptography. In: Jonoska, N., Păun, G., Rozenberg, G. (eds.) *Aspects of Molecular Computing*. LNCS, vol. 2950, pp. 167–188. Springer, Heidelberg (2003)
- [31] Guoan, B.: Fast algorithms for the 2-D discret W transform. *Signal Processing* 74, 297–308 (1999)
- [32] Hartung, F., Girod, B.: Watermarking of uncompressed and compressed video. *Signal Processing* 66, 283–301 (1998)
- [33] Hartung, F., Girod, B.: Digital Watermarking of Raw and Compressed Video. *Digital Compression Technologies and Systems for Video Communications* 2952, 205–213 (1996)
- [34] Hartung, F., Girod, B.: Digital Watermarking of MPEG-2 Coded Video in the Bit-stream Domain. In: *Proceedings of International Conference on Acoustic, Speech and Signal Processing*, vol. 4, pp. 2985–2988 (1997)
- [35] Hartung, F., Kutter, M.: Multimedia Watermarking Techniques. *Proceedings of IEEE* 87(7), 1079–1106 (1999)
- [36] Hartung, F., Su, J.K., Girod, B.: Spread Spectrum Watermarking: Malicious Attacks and Counterattacks. In: *Proc. Of SPIE Security and Watermarking of Multimedia Contents*, vol. 3957, pp. 147–158 (1999)
- [37] Hernandez, J.R., Perez-Gonzales, F.: Statistical Analysis of Watermarking Schemes for Copyright Protection of Images. *Proceedings of IEEE* 87(7), 1142–1166 (1999)
- [38] <http://www.ncbi.nlm.nih.gov>
- [39] Jayant, N., Johnston, J., Safranek, R.: Signal Compression Based on Models of Human Perception. *Proceedings of IEEE* 81(10), 1385–1422 (1993)
- [40] Joseph, J.K., Ruanaidh, O., Pun, T.: Rotation, scale and translation invariant spread spectrum digital image watermarking. *Signal Processing* 66, 303–317 (1998)
- [41] Kahn, D.: *The Codebreakers*. McMillan, New York (1967)
- [42] Katzenbeisser, S., Petitcolas, F.: *Information hiding techniques for steganography and digital watermarking*. Artech House, Boston (2000)
- [43] Kim, S.W., Suthaharan, S., Lee, H.K., Rao, K.R.: Image watermarking scheme using visual model and BN distribution. *Electronics Letters* 35(3), 212–214 (1999)
- [44] Kingsbury, N.G.: The Dual-Tree Complex Wavelet Transform: A new Tehnique for Shift Invariance and Directional Filters. In: *Proc. 8th IEEE DSP Workshop*, Bryce Canyon (1998)
- [45] Kobayaski, M.: Digital Watermarking: Hystorical roots. IBM Research. Apr. Tokyo Res. Lab. Technical Report (1997)
- [46] Langelaar, G.C., Lagendijk, R.L., Biemond, J.: Real time labeling methods for MPEG compressed video. In: *Proc. 18th Symp. on Information Theory in the Benelux*, Voldhoven, NL (1997)
- [47] Langelaar, G.C., Setyawan, I., Lagendijk, R.L.: Watermarking Digital Image and Video Data. *Signal Processing Magazine* 20–46 (2000)
- [48] Low, S., Maxemchuk, N.: Performance comparison of two text marking methods. *IEEE J. Selected Areas Commun.* 16, 561–571 (1998)

- [49] Low, S., Maxemchuk, N., Brasil, J., O' German, L.: Document marking and identification using both line and word shifting. In: Proc. Infocom 1995, Boston (1995)
- [50] Lu, C.S., Huang, S.K., Sze, C.J., Liao, H.Y.M.: A New Watermarking Tehnique for Multimedia Protection. In: Guan, L., Kung, S.Y., Larsen, J. (eds.) Multimedia Image and Video Processing. CRC Press Inc., Boca Raton (1999) (to appear)
- [51] Lu, C.S., Liao, H.Y.M., Huang, S.K., Sze, C.J.: Cocktail Watermarking on Images. In: Pfizmann, A. (ed.) IH 1999. LNCS, vol. 1768, pp. 333–347. Springer, Heidelberg (2000)
- [52] Lu, C.-S., Liao, H.-Y.M., Huang, S.-K., Sze, C.-J.: Highly Robust Image Watermarking Using Complementary Modulations. In: Zheng, Y., Mambo, M. (eds.) ISW 1999. LNCS, vol. 1729, pp. 136–153. Springer, Heidelberg (1999)
- [53] Maes, M., Kalker, T., Linnartz, J.P., Talstra, J., Depovere, G.F., Haitsma: Digital water-marking for DVD video copy protection. Signal Processing Magazine 17(5), 47–57 (2000)
- [54] Major, R., Borda, M., Amadou, K.: Hash functions for spatial image watermarking. In: International Scientific Conference microCAD, Miskolc, Ungaria, pp. 39–44 (2003)
- [55] Maxemchuk, N., Low, S.: Marking text documents. In: Proc. IEEE Int. Conf. Image Processing, Santa Barbara, pp. 13–16 (1997)
- [56] Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
- [57] Nafornita, C., Borda, M., Cane, A.: Wavelet-based digital watermarking using sub-band- adaptative thresholding for still images. In: Proceeding of microCAD, Miskolc, Hungary, pp. 87–91 (2004)
- [58] Nikolaidis, N., Pitas, I.: Robust image watermarking in the spatial domain. Signal Processing 66, 385–403 (1998)
- [59] Petitcolas, F.A.P.: Watermarking Schemes Evaluation. Signal Processing Magazine, 58–67 (2000)
- [60] Petitcolas, F.A.P., Anderson, R.J., Kulu, M.G.: Information hiding - A survey. Proceedings of the IEEE 87(7), 1062–1078 (1999)
- [61] Podlichuk, C.I., Zeng, W.: Image-adaptive watermarking using visual models. IEEE J. Select. Area Commun. 16, 525–539 (1998)
- [62] Roweis, S., Winfree, E., Burgoyne, R., et al.: A sticker based architecture for DNA computation. In: DIMACS series in discrete mathematics and theoretical computer science, vol. 44, pp. 1–30 (1996)
- [63] Schena, M.: Microarray analyses. Wiley-Liss, Chichester (2003)
- [64] Schneier, B.: Applied cryptography: Protocols. In: Algorithms and Source Code in C. John Wiley & Sons, Inc., Chichester (1996)
- [65] Shannon, C.: A Mathematical Theory Of Communication. Bell System Technical Journal 27, 379–423 (1948); Reprinted in Shannon Collected Papers, IEEE Press (1993)
- [66] Shannon, C.: Communication Theory of Secrecy Systems. Bell System Technical Journal 28(4), 656–715 (1949)
- [67] Smith, M.: Station X: The Codebreakers of Bletchley Park. Channel 4 Books (2000)
- [68] Stallings, W.: Cryptography and Network security, Principles and Practice, 2nd edn. Prentice-Hall, Englewood Cliffs (1998)
- [69] StirMark, <http://www.cl.cam.ac.uk/>
- [70] Swanson, M.D., Kobayashi, M., Tewfik, A.H.: Multimedia Data Embedding and Watermarking Technologies. Proceedings of the IEEE 86(6), 1064–1087 (1998)

- [71] Swanson, M.D., Zhu, B., Chau, B., Tewfik, A.H.: Object-Based Transparent Video Wa-termarking. In: IEEE First Workshop on Multimedia Signal Processing, pp. 369–374 (1997)
- [72] Swanson, M.D., Zhu, B., Tewfik, A.H.: Multiresolution scene-based video watermarking using perceptual models. *IEEE J. Select. Area Commun.* 16(4), 540–550 (1998)
- [73] Tanaka, K., Nakamura, Y., Matsui, K.: Embedding secret information into a dithered multilevel image. In: Proc. IEEE Military Commun. Conference, pp. 216–220 (1990)
- [74] Taylor, C., Risca, V., Bancroft, C.: Hiding messages in DNA microdots. *Nature* 399, 533–534 (1999)
- [75] Tewfik, A.H.: Digital Watermarking. *IEEE Signal Processing Magazine* 17(5), 17–18 (2000)
- [76] Tewfik, A.H., Hamdy, K.H.: Digital watermarks for audio signals. In: Proc. EUSIPCO 1996, Trieste, Italy (1996)
- [77] Tirkel, A., van Schyndel, R., Osborne, C.: A two-dimensional watermark. In: Proc. DICTA, pp. 666–672 (1993)
- [78] Tornea, O., Borda, M.: DNA Cryptographic Algorithms. *MediTech Cluj-Napoca* 26, 223–226 (2009)
- [79] U. S. Copyright office Summary, The Digital Millennium Copyright Act of 1998 (December 1998), <http://www.loc.gov/copyright/leg>
- [80] Viterbi, A.J.: CDMA-Principles of spread spectrum communications. Addison – Wesley, London (1996)
- [81] Wade, G.: Watermark Attacks. Research Report, Plymouth (1999)
- [82] Wang, H.: Proving theorems by pattern recognition. *Bell Systems Technical Journal* 40, 1–42 (1961)
- [83] Wang, H.J.M., Su, P.C., Kuo, C.-C.J.: Wavelet-based digital image watermarking. *Optics Express* 3(12), 491–496 (1998)
- [84] Watson, A.T.J.: IBM Research Report. RC 20509 Computer science/ Mathematics (1996)
- [85] Wolfgang, R.B., Delp, E.J.: Overview of image security techniques with applications in multimedia systems. In: Proc. SPIE Int. Conf. Multimedia Networks: Security, Display, Terminals, and Gateways, vol. 3228(4/5), pp. 297–308 (1997)
- [86] Wolfgang, R.B., Podiluck, C.I., Delp, E.J.: Perceptual Watermarks for Digital Images and Video. *Proceedings of IEEE* 87(7), 1108–1126 (1999)
- [87] Wolfgang, R.B., Delp, E.J.: Fragile watermarking using the VW2D watermark. *Proc. Electronic Imaging* 3657, 204–213 (1999)
- [88] Xia, X.G., Boncelet, C.G., Arce, G.R.: Wavelet transform based watermark for digital images. *Optics Express* 3(12), 497–511 (1998)
- [89] Borda, M., Tornea, O., Hodoroaga, T.: Secrete Writing by DNA Hybridization. *Acta Technica Napocensis* 50(2), 21–24 (2009)
- [90] Shparlinski, I.: *Finite Fields: Theory and Computation*. Kluwer Academic Publishers, Boston (1999)
- [91] Lidl, R., Niederreiter, H., Cohn, P.M.: *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, Cambridge (2000)
- [92] Hankerson, D.C., et al.: *Coding Theory and Cryptography*, 2nd edn. Marcel Dekker, New York (2000)
- [93] Johnson, N.F., Duric, Z., Jajodia, S.: *Information Hiding*. Kluwer Academic Publishers, Boston (2001)

Chapter 5

Channel Coding

Motto: *The way towards truth is strewn with errors. Who does not make them, who does not touch them, does not fight with them and finally doesn't obviate them with his own forces, does not reach the truth.*

Constantin Tsatsos

5.1 Shannon Second Theorem (Noisy Channels Coding Theorem)

When transmitting (storing) information via noisy channels (storage media), the noise may alter the signals. This is why protection against this unwanted effect must be taken. This problem occurred since the very beginnings of communications, but it became a huge problem with the development of high speed networks and globalisations of digital communications. The advent of electronic era requires high reliability in data transmission and/or storage, taking into account its impact in economical and social life. The history of error control coding shows how the field was born (mid 20th century) and grown from the one error correcting codes (Hamming codes - 1950) until the most powerful (turbo-codes - 1993), trying to limit technically errors effect in applications.

The answer to the question: how is it possible to achieve error protection when transmitting or storing information? was given by Cl. E. Shannon in his work "A mathematical theory of communication" (1948) and represents what is now known under the name of *Shannon second theorem (noisy channels coding theorem)*. He proved that:

- on any noisy channel of capacity C , a real time transmission can be performed ($D < C$) with an error probability $P(E)$ as small as wanted, using uniform codes with length n , such that:

$$P(E) \leq 2^{-ne(D)} \quad (5.1)$$

where:

- n is the codewords length (uniform encoding)
- $e(\dot{D})$ is a positive function of \dot{D} , completely determined by channel characteristics; it is called *error exponent* (see Fig. 5.1).
- 2 – binary channel

Remarks

- the theorem was given under the assumption of binary channel, but it stands for any alphabet (>2)
- the demonstration of (5.1) is found in [41]

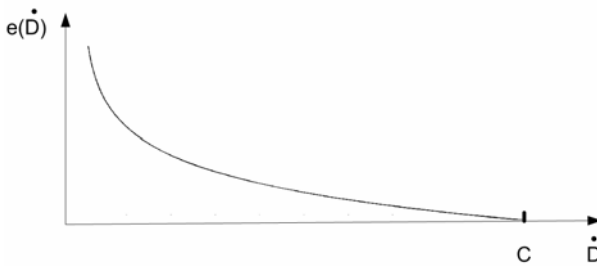


Fig. 5.1 Error exponent graphic

Interpretations of Shannon second theorem

- the theorem underlines a very surprising fact: no matter how noisy the channel is, a transmission with an error probability however small is always possible
- without giving any algorithms, the theorem shows the ways to obtain $P(E)$ however small:
 - transmission at low rate: in this case (see Fig. 5.1) if $e(\dot{D})$ increases, $P(E)$ decreases; this method is not used in practice because the channel is not efficiently used ($\dot{D} \ll C$);
 - using large codewords (great n), which means deliberately introducing redundancy before transmission; this method is used in practice for error protection: redundant codes for error protection - *error control codes*
- in real transmission channels (noisy), the error probability $P(E)$ can not be 0, because it means $n \rightarrow \infty$, meaning infinite bandwidths (see 2.8.5 – Interpretations of Shannon limit).

A simple example will be given to explain the necessity of introducing redundancy in order to detect and correct errors.

Example 5.1

Let us consider $S = \{s_1; s_2\}$ an equally probable binary source. Considering the transmission in a binary symmetric channel with p known, we shall analyze the two following cases:

- *Non-redundant coding*: it follows that the codewords length is $n = m = 1$, where m designate the number of *information symbols*, necessary to encode the information of the source S and so we have the code $C_1 = \{0;1\}$. Transmitting over a binary symmetric channel BSC, we receive the same symbols $\{0; 1\}$ without knowing if one of them is erroneous; the error probability in this case will be:

$$P_1(E) = p.$$

- *Redundant coding*: instead of transmitting one symbol, we transmit $n = 3$ symbols, introducing $k = n - m = 3 - 1 = 2$ *redundant symbols (control symbols)*, the code becoming: $C_2 = \{000;111\}$

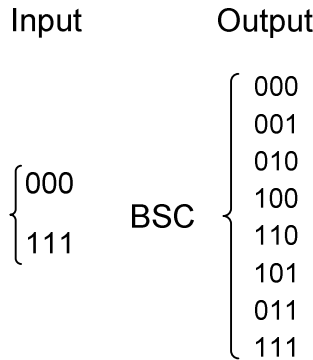


Fig. 5.2 Input/output representation of a BSC obtained for C_2 .

When transmitting over a BSC we obtained the situation shown in Fig. 5.2. From 8 possible output sequences only 2 correspond to the codewords and the remaining 6 perform the error detection. If we chose as rule the *majority criterion* (that means we consider reliable sequences those containing two “0”s and a “1” or two “1”s and one “0” corresponding to the sequences 000 and 111), the correction is also possible.

Obviously, the decisions taken at the receiver are exposed to a risk that can also be evaluated. Even when receiving the codewords 000 and 111 there is a risk that all the three bits are erroneous. For independent errors, this risk becomes $p_3 = p^3$. Similarly the probability of two errors in a word of length 3 is:

$$p_2 = C_n^2 p^2 (1 - p) = 3p^2 (1 - p).$$

The total error probability after decoding becomes in this case:

$$P_2(E) = p_2 + p_3 = 3p^2 (1 - p) + p^3 = p^2 (3 - 2p) \cong 3p^2$$

The initial risk has decreased from $P_1(E) = p$ to $P_2(E) = 3p^2$.

A second question rises: which is the price paid to obtain an error probability however small on the account of the redundancy increase?

The answer to this question may be obtained reasoning as follows:

If we want to keep unmodified the information source decision rate (\dot{D}_i):

$$\dot{D}_i = \frac{1}{T_{bi}} \quad (5.2)$$

where T_{bi} is the information bit duration, than adding k redundant symbols (Fig. 5.3) we obtain:

$$nT_{bc} = mT_{bi} \quad (5.3)$$

where T_{bc} is the coded bit duration.

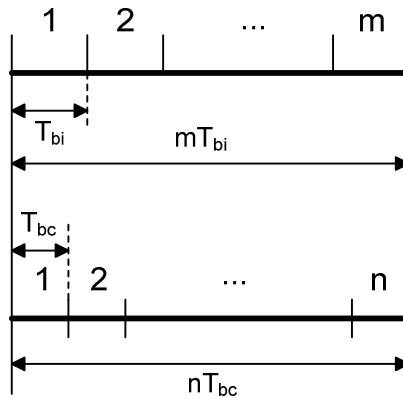


Fig. 5.3 Illustration of the relations between information bits and coded ones

From (5.3) we obtain:

$$T_{bc} = \frac{m}{n} T_{bi} \quad (5.4)$$

The ratio:

$$R = \frac{m}{n} \quad (5.5)$$

is named the *coding rate*.

Equation (5.3) can be written also as:

$$\dot{D}_c = \frac{1}{T_{bc}} = \frac{\dot{D}_i}{R} > \dot{D}_i \quad (5.6)$$

so, encoding and keeping $\dot{D}_i = ct$, we obtain an increase of the coded decision rate \dot{D}_c . Knowing that \dot{D}_c and the required bandwidth B are directly proportional (see relation (2.82)), it results that the price paid for error protection when $\dot{D}_i = ct$, is an increase in the transmission bandwidth (the storing space) and consequently an increased noise at the receiver. When the bandwidth enhancement is not possible, some coding and modulation procedures for bandwidth compression (m-ary phase modulations, trellis modulation [42]) are used.

Remark

The entire discussion in this paragraph has been done under constant power assumption at transmission and without analyzing the modulation system effect and that of the input signal waveform.

In a complete information transmission (storage) system, the channel coding (C_c) block is located as shown in Fig 5.4.

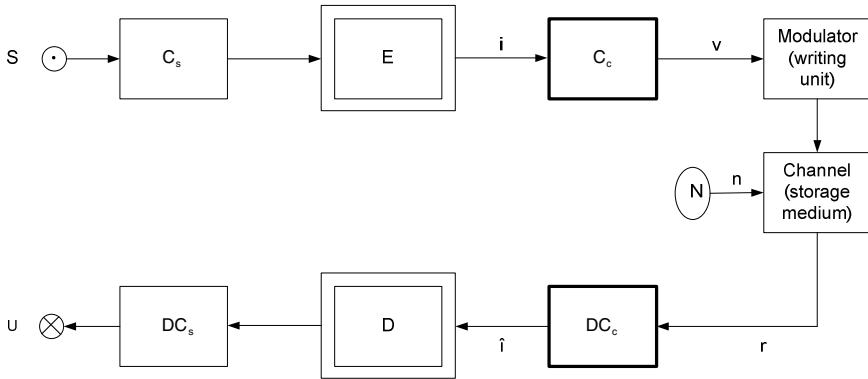


Fig. 5.4 Location of channel coding block in a complete transmission (storage) system.

5.2 Error Control Strategies

Error control in transmission or storage can be achieved mainly in three ways:

- *Error detection*: in this case the system requires a return channel through which the transmitter is informed about error detection and a repetition is necessary (*ARQ – automatic repeat request*)
- *Forward error correction (FEC)* – the error correcting code automatically corrects an amount of errors. The corresponding system is a unidirectional one (Fig. 5.4.); this type of transmission is used in storage systems, space communications etc.

- *Error correction and detection (ARQ hybrid systems)*: the error correcting code will correct the erroneous combinations according to its correcting capacity, the rest of erroneous combinations being corrected by the ARQ system; these systems are used in radio transmissions (satellite, mobile communications)

The most frequently used ARQ procedures are the three ones (Fig. 5.5):

- *Stop and wait (SW) system*: the transmitter sends a code sequence to the receiver and waits for the positive confirmation signal-acknowledge (ACK), which means that no errors were detected; when receiving the ACK signal, the transmitter sends the next sequence. When receiving a negative confirmation – not acknowledge (NAK) corresponding to an erroneous block, the transmitter repeats the NAK block until it receives a positive confirmation (ACK). This is a *half-duplex-type communication*.

This system is extremely simple and it is used in numerous data transmission system such as binary symmetrical communication protocol (BISYNC). Even if it is extremely simple, this system is inefficient because of the break between the transmission and reception of the confirmation.

- *Go-back N blocks system (GBN)* corresponds to a continuous blocks transmission, therefore full duplex communication. The transmitter does not wait for the transmitted block confirmation. This confirmation is received after a number of N blocks. During this interval the transmitter has already transmitted another N-1 blocks. When receiving a NAK, the transmitter returns to the not acknowledged block and repeats this block and another N-1 blocks.

This system is more efficient than SW system and it is less costly. The ARQ system is used in SDLC (Synchronous Data Link Control) and ADCCP (Advanced Data Communication Control Procedure). The inefficiency occurs because of numerous correct blocks that must be repeated.

- *Selective repeat system (SR)* corresponds to a continuous transmission (full duplex communication) just like GBN system, with the difference that, for the former, only the negatively confirmed (NAK) blocks are repeated. This kind of system is the most efficient from the three presented, but the most expensive, in implementation; it is used in satellite communications.

An ARQ system makes decoding errors if it accepts received words affected by non-detectable errors $P(E) = P_{en}$.

The performance of an ARQ system can be estimated by its *total efficiency* (η) defined as the ratio between the total number of information bits accepted by the receiver in time unit and the total number of transmitted bits in a time unit.

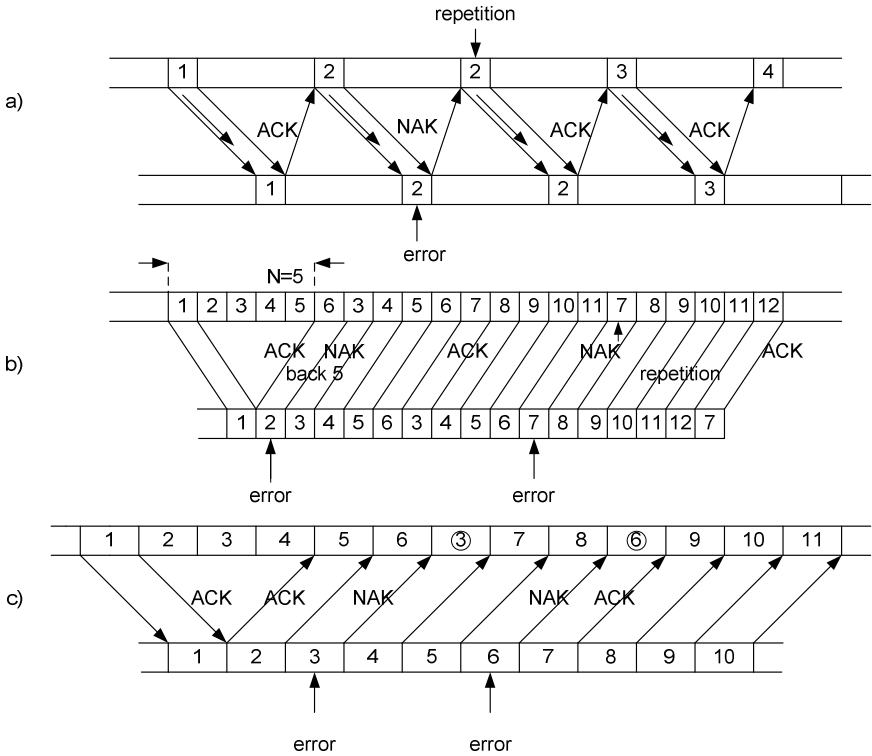


Fig. 5.5 ARQ systems for $N = 5$: a) SW(Stop and Wait); b) GBN(go back N); c) SR(selective repeat).

Remarks

- all the three ARQ methods have the same $P(E)$, but different efficiencies.
- expression of ARQ systems efficiency are determined in [28] under an error-free feedback-channel assumption and in [3] for the case of feedback-channel is noisy too.

Comparison between error control strategies

A detailed analysis of this issue is made in [28].

Compared to FECs, ARQ systems are much simpler. ARQ systems are adaptive, meaning that the information is repeated every time when errors occur. If the channel is very noisy, repetitions became very frequent and justify, in this case, the use of hybrid ARQ + FEC systems.

5.3 Classification of Error Control Codes

Classification of error control codes can be done using many criteria; some of them are listed below:

- according to the nature of the processed information:
 - *block codes*: the information at the input is divided into blocks of m symbols to which, by encoding, k control symbols are added resulting an encoded block of length n .
 - *continuous (convolutional) codes*: the information is processed continuously
- according to channel errors type:
 - *codes for independent error control*
 - *codes for burst error*
- according to the control strategy:
 - *error detecting codes (ARQ)*
 - *error correcting codes (FEC)*
- according to the possibility of separating the information symbols from the control ones:
 - *separable codes*, in which the redundant symbols can be separated from the information symbols
 - *non-separable codes (with implicit redundancy)* in which the separation is not possible, for example the (m, n) codes
- according to the possibility of having or not, the information and control symbols in clear in a codeword, the separable codes can be classified in:
 - *systematic codes*: $\mathbf{v} = [\mathbf{i} \mathbf{c}]$, where \mathbf{i} is the matrix of information symbols and \mathbf{c} the control symbol matrix
 - *non-systematic codes*: $\mathbf{v} = [u^{(1)} \dots u^{(n)}]$ the information and the control symbols are not given in clear in the encoded structure.

5.4 Representation of Binary Code Sequences

The most frequent representations used for binary code sequences are matrix, vector, polynomial and geometrical representations.

Matrix representation is in fact a matrix containing all the code words, with the exception of the all zero components.

If M is the number of n length codewords, than the matrix in which the whole code is included is:

$$\mathbf{C} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{M1} & a_{M2} & \dots & a_{Mn} \end{bmatrix} \Rightarrow \begin{matrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \dots \\ \mathbf{v}_M \end{matrix}$$

where $a_{ij} \in \{0;1\}$ for binary codes.

The matrix representation allows a compact writing by selecting the lines (codewords) that are linearly independent; all the other codewords \mathbf{v}_i can be obtained as linear combinations of the linearly independent lines.

Vector representation of a code is based on the fact that the set of all n -length words forms a vector space \mathbf{V}_n . Each sequence of length n is represented by a vector:

$$\mathbf{v} = (a_1 \ a_2 \ \dots \ a_n)$$

where $a_i \in \{0;1\}$ for binary codes.

The vector space \mathbf{V}_n has, in the binary case, 2^n code vectors formed by “0”s and “1”s. A group of vectors that have a common propriety form a vector subspace \mathbf{V}_m with $m < n$. Such a mathematical model applied for a code allows, when studying the codes, to use the vector spaces properties. We remind some basic notions from the vector space theory used to study linear codes, the most important codes used in communication systems (see also Appendix A).

The vector space order is given by the number of linearly independent vectors; they form a base of the vector space if any vector which belongs to the space can be expressed as a linear combination of linearly independent vectors.

There is a strong dependency between matrix and vector representation as the lines of the matrix can be vectors in a vector space.

The *polynomial representation*: a word of length n : a_0, \dots, a_{n-1} can be represented as an $(n-1)$ or smaller degree polynomial with the unknown x :

$$v(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \dots + a_1x + a_0$$

where $a_i \in \{0;1\}$ in the binary codes case; the power of the unknown x is used to locate symbol a_i in the sequence.

The polynomial representation allows using, when studying codes, the proprieties of algebraic structures built in polynomials theory (see Appendix A – finite fields).

Geometric representation: every word of length n can be considered as a point, which defines the peaks of a geometrical shape in a n -dimensional space. It allows using a series of well known proprieties of geometric figures to the codes.

5.5 Parameters of Detecting and Error Correcting Codes

In what follows we will define parameters that allow us to compare the performances of error detecting and correcting codes.

Redundancy

As it was shown in Shannon second theorem, error protection can be achieved only using redundancy. When coding, redundancy can be defined in absolute or relative value, for separable and non-separable codes.

- *absolute redundancy* R_a : for separable codes it represents the number of redundant symbols from a codeword

$$R_a = k[\text{bits}] \quad (5.7)$$

- *relative redundancy* R_r : for separable codes it is the ratio between the number of redundant symbols k and the length of code word n :

$$R_r = \frac{k}{n} \Leftrightarrow R_r = \frac{k}{n} 100[\%] \quad (5.8)$$

For *non-separable codes* (implicit redundancy codes) the two parameters can be defined as:

$$R_a = \text{ld}[\text{total no. of } n\text{-length sequences}] - \text{ld}[\text{no. of codewords}] \quad (5.9)$$

$$R_r = \frac{R_a}{\text{ld}[\text{total number of } n\text{-length sequences}]} \quad (5.10)$$

Remark

Equations (5.9) and (5.10) are more general than (5.7) and (5.8), the last two being obtained for a particular case of the former equations. The reader is invited to obtain R_a and R_r for separable codes, starting from the equivalent definitions of the non-separable ones.

Detection and correction capacity

The ratio of detectable/correctable errors from the total number of possible errors is estimated using the *detection/correction capacity* C_d/C_c defined as:

$$C_d = \frac{\text{number of detectable erroneous sequences}}{\text{total number of erroneous sequences}} \quad (5.11)$$

$$C_c = \frac{\text{number of correctable erroneous sequences}}{\text{total number of erroneous sequences}} \quad (5.12)$$

Hamming weight and code distance

- *Hamming weight* w of a word \mathbf{v} is defined by the number of non-zero symbols of that word.

Example

$$\mathbf{v} = 1011011 \Rightarrow w(\mathbf{v}) = 5$$

- *Hamming distance* between two word \mathbf{v}_i and \mathbf{v}_j denoted by $d(\mathbf{v}_i, \mathbf{v}_j)$ is defined as the number of positions for which \mathbf{v}_i and \mathbf{v}_j differ:

$$d(\mathbf{v}_i, \mathbf{v}_j) = \sum_{k=1}^n a_{ki} \oplus a_{kj} \quad (5.13)$$

where $\mathbf{v}_i = (a_{i1}, a_{i2}, \dots, a_{in})$, $\mathbf{v}_j = (a_{j1}, a_{j2}, \dots, a_{jn})$ and \oplus is modulo two summation.

Example

$$\mathbf{v}_i = (1011011)$$

$$\mathbf{v}_j = (0111101)$$

The distance between \mathbf{v}_i and \mathbf{v}_j is:

$$\begin{aligned} d(\mathbf{v}_i, \mathbf{v}_j) &= (1 \oplus 0) + (0 \oplus 1) + (1 \oplus 1) + (1 \oplus 1) + (0 \oplus 1) + (1 \oplus 1) = \\ &= 1 + 1 + 0 + 0 + 1 + 1 + 0 = 4 \end{aligned}$$

For linear block codes modulo two summation of two binary words \mathbf{v}_i and \mathbf{v}_j is another word with “1”s on the positions in which \mathbf{v}_i and \mathbf{v}_j differ.

Theorem

Hamming distance between two words \mathbf{v}_i and \mathbf{v}_j equals the weight of the modulo two summation of the given words:

$$d(\mathbf{v}_i, \mathbf{v}_j) = w(\mathbf{v}_i \oplus \mathbf{v}_j) = w(\mathbf{v}_k) \quad (5.14)$$

For a code C having M codewords the *code distance* d is defined as the minimum distance between any two codewords:

$$d = \min d(\mathbf{v}_i, \mathbf{v}_j), \forall \mathbf{v}_i, \mathbf{v}_j \in C \quad (5.15)$$

According to (5.14) the distance between two codewords equals to another codeword weight; therefore it results that the code distance d equals the minimum weight of that code:

$$d = w_{\min} \quad (5.16)$$

Remarks

- equation (5.16) is very useful in practice due to the fact that it is very easy to compute the weights
- the all zero codewords are not taken into consideration when calculating the weights and code distances.

5.6 Maximum Likelihood Decoding (MLD)

At the receiver the decoder must decide, based on the received sequence \mathbf{r} , which was the transmitted sequence and so it estimates the transmission of $\hat{\mathbf{v}}$ based on the reception. If $\hat{\mathbf{v}} = \mathbf{v}$ (code sequence), then there are no decoding errors. If the estimated sequence $\hat{\mathbf{v}}$ differs from the transmitted one \mathbf{v} the decoding is erroneous. The probability of having errors after decoding, when \mathbf{r} is received, $p(\mathbf{E}/\mathbf{r})$, is:

$$p(\mathbf{E}/\mathbf{r}) =: p(\hat{\mathbf{v}} \neq \mathbf{v}/\mathbf{r}) \quad (5.17)$$

The decoder error probability can also be expressed as:

$$p(\mathbf{E}) = \sum_{\mathbf{r}} p(\mathbf{E}/\mathbf{r})p(\mathbf{r}) \quad (5.18)$$

derived as marginal probability from the joint probability of (\mathbf{E}, \mathbf{r}) :

$$p(\mathbf{E}, \mathbf{r}) = p(\mathbf{E}/\mathbf{r})p(\mathbf{r})$$

According to (5.18), we will have a minimum decoding error probability, if $P(\mathbf{E}, \mathbf{r})$ is minimum, $p(\mathbf{r})$ being independent of the coding process. Due to the fact that the minimization of $p(\mathbf{E}/\mathbf{r})$ is equivalent to the maximization of $p(\mathbf{v} = \mathbf{v}/\mathbf{r})$ it results that $p(\mathbf{E})$ min can be obtained for $p(\mathbf{v}/\mathbf{r})$ max; $p(\mathbf{v}/\mathbf{r})$ can be expressed, according to Bayes formula:

$$p(\mathbf{v}/\mathbf{r}) = \frac{p(\mathbf{v})p(\mathbf{r}/\mathbf{v})}{p(\mathbf{r})} \quad (5.19)$$

It results that the maximization of $p(\mathbf{v}/\mathbf{r})$ is equivalent to the maximization of $p(\mathbf{r}/\mathbf{v})$, in the hypothesis of equally probable codewords. For a memory-less channel, each received symbol depends only on the transmitted one; it follows:

$$p(\mathbf{r}/\mathbf{v}) = \prod_{i=1}^n p(r_i/v_i) \quad (5.20)$$

where r_i and v_i , are the received/transmitted word components. The decoder that estimates the received sequence by maximizing (5.20) is called MLD, i.e. *maximum likelihood decoder*.

$$\max p(\mathbf{r}/\mathbf{v}) = \max \prod_{i=1}^n p(r_i/v_i) \quad (5.21)$$

where v_i are the components of \mathbf{v} .

Due to the fact that $(\log_2 x)$ is a monotone increasing function of x , the maximization of (5.21) is equivalent to the maximization of the logarithm:

$$\max \log_2(\mathbf{r}/\mathbf{v}) = \max \sum_i p(r_i/v_i) \quad (5.22)$$

Remark

- if the codewords are not equally probable, the MLD decoder is not always optimal, in equation (5.20) $p(r_i/v_i)$ being weighted with $p(\mathbf{v})$; in numerous applications $p(\mathbf{v})$ is not known; this is the reason why, the MLD decoding remains in practice the best, so optimal.

For a BSC for which $p(r_i/v_i) = \begin{cases} p, & \text{if } r_i \neq v_i \\ 1-p, & \text{if } r_i = v_i \end{cases}$ for an n -length word, the

equation (5.22) becomes:

$$\begin{aligned} \max[\log_2 p(\mathbf{r}/\mathbf{v})] &= \max\{d(\mathbf{r}, \mathbf{v}) \log_2 p + [n - d(\mathbf{r}, \mathbf{v})] \log_2(1-p)\} = \\ &= \max[d(\mathbf{r}, \mathbf{v}) \log_2 \frac{p}{1-p} + n \log_2(1-p)] \end{aligned} \quad (5.23)$$

Because $\log_2 \frac{p}{1-p} < 0$ for $p < 1/2$ and $n \cdot \log_2(1-p)$ is constant for any \mathbf{v} , the

MLD decoding rule for a BSC estimates $\hat{\mathbf{v}}$ as that word \mathbf{v} that minimizes $d(\mathbf{r}, \mathbf{v})$ and that is why the *MLD decoder is also called minimum distance decoder*.

$$\max [\log_2 p(\mathbf{r}, \mathbf{v})] = \min[d(\mathbf{r}, \mathbf{v})] \quad (5.24)$$

Consequently, a MLD decoder determines the distance between \mathbf{r} and all the possible codewords \mathbf{v}_i and selects \mathbf{v}_i for which $d(\mathbf{r}, \mathbf{v}_i)$ is minimum ($i = \overline{1, M}$). If this minimum is not unique, choosing between several \mathbf{v}_i words becomes arbitrary.

Fig 5.6 shows the minimum distance MLD principle. Let us consider \mathbf{v}_1 and \mathbf{v}_2 as two codewords for which $d(\mathbf{v}_1, \mathbf{v}_2) = 5$.

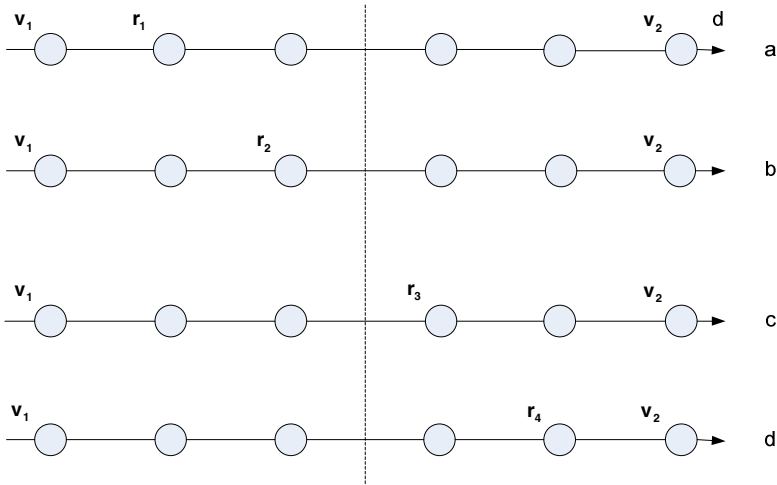


Fig. 5.6 Minimum distance decoder principle ($d=5$).

Let us consider four cases a, b, c, and d, corresponding to the reception of the words $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4$ located at distance 1, 2, 3 and 4 from \mathbf{v}_i . For case a, corresponding to one error in \mathbf{v}_1 the decoder will decide that \mathbf{r}_1 comes from \mathbf{v}_1 due to the fact that $d(\mathbf{v}_1, \mathbf{r}_1) = 1 < d(\mathbf{v}_2, \mathbf{r}_2) = 4$. The decoding would have been erroneous if \mathbf{r}_1 had come from \mathbf{v}_2 , by setting errors on 4 symbols. For case b, corresponding to two errors in \mathbf{v}_1 , the decoder will decide that \mathbf{r}_2 comes from \mathbf{v}_1 as $d(\mathbf{v}_1, \mathbf{r}_2) = 2 < d(\mathbf{v}_2, \mathbf{r}_2) = 3$; decoded errors occur if \mathbf{r}_2 had come from \mathbf{v}_2 being erroneous by three symbols. For case c, the decoder will decide that \mathbf{r}_3 comes from \mathbf{v}_2 by the occurrence of two errors: $d(\mathbf{v}_2, \mathbf{r}_3) = 2 < d(\mathbf{v}_1, \mathbf{r}_3) = 3$. For case d, the decoder will decide that \mathbf{r}_4 comes from \mathbf{v}_2 under the assumption of one error: $d(\mathbf{v}_2, \mathbf{r}_4) = 1 < d(\mathbf{v}_1, \mathbf{r}_4) = 4$. For the last cases the same discussion is possible as in cases a and b. Fig. 5.6 shows that error detection is possible when $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4$ are received corresponding to errors on 1, 2, 3 or 4 positions on \mathbf{v}_1 or \mathbf{v}_2 . Non-detectable errors occur when the reception is erroneous on 5 positions.

This example shows that the code distance is a parameter able to indicate the code detection and correction capacity. In [28] it is demonstrated that:

- the necessary and sufficient condition for a code to correct maximum t errors is:

$$d \geq 2t + 1 \quad (5.25)$$

- the necessary and sufficient condition for a code to detect maximum e errors is:

$$d \geq e + 1 \quad (5.26)$$

- the necessary and sufficient condition for a code to simultaneously correct maximum t errors and detect maximum e errors is:

$$d \geq t + e + 1, e > t \quad (5.27)$$

Remark

- Correction implies detection first; for example, for $d = 7$ we may have:

e (detection)	t (correction)
3	3
4	2
5	1
6	0

- if three errors occur, all of them can be detected and corrected
- if five errors occur, all of them can be detected but only one of them can be corrected.

The same parameter d can be used to estimate the *erasure correction capacity* [42]. For *erasure channels* the channel output alphabet has a supplementary symbol y_3 called *erasure symbol*. The moment the decoder decides that a symbol is an erasure symbol, even though the correct value is unknown, its location is known, subsequently the correction of the erasure errors must be simpler than for other errors (for the general case both the position and value must be known). It can be demonstrated that for correcting a number of s erasures the following condition must be fulfilled [42]:

$$d \geq s + 1 \quad (5.28)$$

The condition for simultaneous correction of t errors and s erasures is:

$$d \geq 2t + s + 1 \quad (5.29)$$

The simultaneous correction of s erasure errors and t errors is performed as follows. At the beginning, the s erased positions are replaced by “0” and then the codeword is decoded with standard methods. The next step is to replace the s erased positions with “1” and repeat the decoding algorithm. From the two decoded words we select the one corresponding to the minimum number of corrected errors except for the s erased positions.

Example 5.2

Let us consider the code: 0000, 0101, 1010, 1111. It can easily be determined that $d = 2$; according to (5.28) it follows that $s = 1$ erasure may be corrected.

Let us transmit the word 0101. At the receiver we assume that an erasure symbol $\mathbf{r} = x101$ is detected.

Replacing x with 0 we obtain $\mathbf{r}_1 = 0101 = \mathbf{v}_2$, therefore $d(\mathbf{r}_1, \mathbf{v}_2) = 0$. Replacing x with 1 we obtain $\mathbf{r}_2 = 1101 \in C$; it follows $d(\mathbf{r}_2, \mathbf{v}_2) = 1$, $d(\mathbf{r}_2, \mathbf{v}_3) = 3$, $d(\mathbf{r}_2, \mathbf{v}_4) = 1 \Rightarrow \hat{\mathbf{r}} = \mathbf{v}_2$.

5.7 Linear Block Codes

Short time after Shannon gave his second theorem, the first error protection codes were invented: the linear block-codes. The main steps in the history of evolution of these codes are: 1950 – R. Hamming and M. Golay introduced the systematic linear codes, 1957 –1960 – D. Slepian contributed to a unitary theory for linear codes; since 1960 many papers studied in detail the linear codes and proposed a series of practical codes.

The binary codes, those having the symbols in GF(2), will be presented in what follows [2], [28].

5.7.1 Linear Block Codes: Definition and Matrix Description

As shown in 5.3 - when dealing with block codes - the information originating in a binary source is divided into m bits blocks denoted with \mathbf{i} (*information block*); to the \mathbf{m} information symbols we add \mathbf{k} redundant (*control*) symbols according to a certain coding law, thus giving a codeword \mathbf{v} of length n ; it follows that:

$$n = m + k \quad (5.30)$$

The number of messages that can be encoded and also the number of codewords, for such a structure, is:

$$M = 2^m \quad (5.31)$$

From block codes, the linear structures are very important for practical applications.

We call a *linear block code* (n,m) the n -length code for which the 2^m (for binary codes) codewords form an m -dimensional subspace \mathbf{C} of the n -dimensional space \mathbf{V}_n that contains all the words of n bits (with coefficients in GF(2)).

A block code is *linear* if and only if the modulo 2 summation of two codewords is still a code word (see the properties of vector spaces – Appendix A8).

Defining such a structure it results that the n -dimensional vector space \mathbf{V}_n containing the set of distinct combinations that can be generated with n bits (2^n) is divided into two distinct sub-sets:

- \mathbf{C} – the set of codewords, having 2^m elements
- \mathbf{F} – the set of fake words, containing $2^n - 2^m$ elements

Due to the fact that the linear code $\mathbf{C}(n,m)$ is an m -dimensional subspace of \mathbf{V}_n , the entire set of codewords can be generated the linear combinations of the m linear independent vectors belonging to \mathbf{C} . These linear independent codevectors denoted with $\mathbf{g}_i, i = \overline{1, m}$ can be written as a matrix; these vectors form the *code generating matrix* $\mathbf{G}_{[m \times n]}$:

$$\mathbf{G} = \begin{bmatrix} g_{11} & g_{12} & \dots & g_{1n} \\ g_{21} & g_{22} & \dots & g_{2n} \\ \dots & \dots & \dots & \dots \\ g_{m1} & g_{m2} & \dots & g_{mn} \end{bmatrix} \Rightarrow \begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ \dots \\ \mathbf{g}_m \end{bmatrix} \tag{5.32}$$

where $g_{ij} \in GF(2)$, meaning they are binary symbols.

Subsequently, *the encoding law* is given by:

$$\mathbf{v} = \mathbf{iG} = \begin{bmatrix} i_1 & i_2 & \dots & i_m \end{bmatrix} \begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ \dots \\ \mathbf{g}_m \end{bmatrix} = i_1\mathbf{g}_1 + i_2\mathbf{g}_2 + \dots + i_m\mathbf{g}_m \tag{5.33}$$

To obtain a systematic structure:

$$\mathbf{v}' = [\mathbf{i} \quad \mathbf{c}] \tag{5.34.a}$$

or

$$\mathbf{v}'' = [\mathbf{c} \quad \mathbf{I}] \tag{5.34.b}$$

the generating matrix \mathbf{G} must have one of the two *canonical forms*:

$$\mathbf{G}' = [\mathbf{I}_m \quad \mathbf{P}] \tag{5.32.a}$$

$$\mathbf{G}'' = [\mathbf{P} \quad \mathbf{I}_m] \tag{5.32.b}$$

In this case the encoding relation, (5.33), becomes:

$$\begin{aligned} \mathbf{v}' &= [\mathbf{i} \quad \mathbf{c}] = \mathbf{iG}' = \\ &= \begin{bmatrix} i_1 & i_2 & \dots & i_m \end{bmatrix} \begin{bmatrix} \mathbf{I}_m & \mathbf{P}_{[kxk]} \\ \left[\begin{array}{cccc|cccc} 1 & 0 & \dots & 0 & p_{11} & p_{12} & \dots & p_{1k} \\ 0 & 1 & \dots & 0 & p_{21} & p_{22} & \dots & p_{2k} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & p_{m1} & p_{m2} & \dots & p_{mk} \end{array} \right] \end{bmatrix} = \tag{5.33.a} \\ &= \begin{bmatrix} i_1 & i_2 & \dots & i_m & \sum_{j=1}^m i_j p_{j1} & \sum_{j=1}^m i_j p_{j2} & \dots & \sum_{j=1}^m i_j p_{jk} \\ & & & & \downarrow & \downarrow & & \downarrow \\ & & & & c_1=f_1(i_j) & c_2=f_2(i_j) & & c_3=f_3(i_j) \end{bmatrix} \end{aligned}$$

Looking at this equation we notice that the m information symbols are found unchanged in the codeword \mathbf{v} structure and that the k control symbols (assigned to c_i) are linear combinations of information symbols; $\sum_{j=1}^m$ means modulo-two sum.

$$c_i = f_i(i_j), \quad i = \overline{1, k}, \quad j = \overline{1, m} \quad (5.35)$$

Equations (5.35) are known as *encoding equations* or *parity-check equations*.

Example 5.3

Let us consider the code $C(5,3)$ with the encoding law:

$$c_1 = i_1 \oplus i_2$$

$$c_2 = i_2 \oplus i_3$$

Taking into consideration the encoding equations, we shall determine the canonical form of the generating matrix \mathbf{G}' as follows:

i_1	i_2	i_3		c_1	c_2	\mathbf{g}_i
1	0	0		1	0	$\mathbf{g}_1 = \mathbf{v}_1$
0	1	0		1	1	$\mathbf{g}_2 = \mathbf{v}_2$
0	0	1		0	1	$\mathbf{g}_3 = \mathbf{v}_3$

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

From the linear combinations of the three linear independent vectors of the code \mathbf{g}_i we obtain the other four non-zero codewords:

$$\mathbf{v}_4 = \mathbf{g}_1 \oplus \mathbf{g}_2 = [1 \ 1 \ 0 \ 0 \ 1]$$

$$\mathbf{v}_5 = \mathbf{g}_1 \oplus \mathbf{g}_3 = [1 \ 0 \ 1 \ 1 \ 1]$$

$$\mathbf{v}_6 = \mathbf{g}_2 \oplus \mathbf{g}_3 = [0 \ 1 \ 1 \ 1 \ 0]$$

$$\mathbf{v}_7 = \mathbf{g}_1 \oplus \mathbf{g}_2 \oplus \mathbf{g}_3 = [1 \ 1 \ 1 \ 0 \ 0]$$

From the binary vector spaces properties (Appendix A.8) we know that if we consider a space \mathbf{C} of dimension m , then always exists a null (orthogonal) space \mathbf{C}^* of dimension $k = n - m$ such that a codeword $\mathbf{v} \in \mathbf{C}$ is orthogonal in \mathbf{C}^* . The linear independent k vectors belonging to the null space \mathbf{C}^* can be put in a matrix $\mathbf{H}_{[k \times n]}$ named *control matrix* (*parity check matrix*):

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & \dots & h_{1n} \\ h_{21} & h_{22} & \dots & h_{2n} \\ \cdot & \cdot & \dots & \cdot \\ h_{k1} & h_{k2} & \dots & h_{kn} \end{bmatrix} = [\mathbf{h}_1 \quad \mathbf{h}_2 \quad \dots \quad \mathbf{h}_n] \quad (5.36)$$

where $h_{ij} \in \text{GF}(2)$ and $\mathbf{h}_i, i = \overline{1, n}$ are the control matrix columns.

Knowing that \mathbf{C} and \mathbf{C}^* are orthogonal spaces, it results that matrices \mathbf{G} and \mathbf{H} are also orthogonal:

$$\mathbf{GH}^T = \mathbf{HG}^T = \mathbf{0} \quad (5.37)$$

Remark

- Superscript T refers to “*transposed matrix*”.

When dealing with systematic linear codes, \mathbf{H} matrix also requires the correspondent canonical forms:

$$\mathbf{H}' = [\mathbf{P}^T \ \mathbf{I}_k] \quad (5.37.a)$$

$$\mathbf{H}'' = [\mathbf{I}_k \ \mathbf{P}^T] \quad (5.37.b)$$

The encoding equation (5.33) valid in the space \mathbf{C} becomes in \mathbf{C}^* :

$$\mathbf{H}\mathbf{v}^T = \mathbf{0} \quad (5.38)$$

5.7.2 Error Syndrome

Let us consider the reception of word \mathbf{r} , which is erroneous under the assumption of additive noise:

$$\mathbf{r} = \mathbf{v} \oplus \mathbf{e} \quad (5.39)$$

where \mathbf{v} is the transmitted codeword and \mathbf{e} is the error word; \oplus is the modulo two summation and indicates additive errors.

In the case of matrix representation, \mathbf{e} can be written as follows:

$$\mathbf{e} = [e_1 \ e_2 \ \dots \ e_n] \quad (5.40)$$

where e_i is “1” if on the position i appears an error and “0” if no error appears on the i position (obviously for binary codes).

At the receiver, the encoding law is checked:

$$\mathbf{H}\mathbf{r}^T = \mathbf{S} \quad (5.41)$$

where \mathbf{S} is the *syndrome*, in fact a column matrix with k elements.

Replacing \mathbf{r} with (5.39), equation (5.41) can be written as:

$$\mathbf{S} = \mathbf{H}(\mathbf{v} \oplus \mathbf{e})^T = \mathbf{H}\mathbf{v}^T \oplus \mathbf{H}\mathbf{e}^T = \mathbf{H}\mathbf{e}^T \quad (5.42)$$

Remarks

- One may notice that the syndrome does not depend on the transmitted word \mathbf{v} , but only on the errors \mathbf{e} introduced by the channel.
- No errors or undetectable errors implies $\mathbf{S} = \mathbf{0}$. In the case when \mathbf{v} is transformed by errors into another codeword, the encoding equation is still fulfilled and the error cannot be detected. There are a number of $2^m - 1$ cases of *undetectable* errors corresponding to other codewords than those transmitted over the channel.

- If $\mathbf{S} \neq \mathbf{0}$ the errors are detected and, in the case of ARQ systems, the retransmission is requested. If the goal is to correct errors (FEC) than, in the binary codes case, it is necessary to determine the erroneous positions from \mathbf{S} . The distinct and non zero syndromes number (the all zero syndrome corresponds to the absence of errors) is $2^k - 1$; it follows that from the total number of possible errors $(2^n - 1)$ only $(2^k - 1)$ can be corrected.

5.7.3 Dimensioning of Error Correcting Linear Block Codes

Dimensioning a block code means, in fact, to determine the measures m , k and n necessary for encoding a source of M messages in order to correct t errors.

The number of information symbols m is determined by the number of source messages M ; for binary codes m is found from the following inequality:

$$2^m \geq M \quad (5.43)$$

The *necessary condition*, however, not sufficient, for generating a t error correcting code is:

$$2^k \geq \sum_{i=0}^t C_n^i \quad (5.44)$$

This inequality is also known as *Hamming boundary*.

The *sufficient condition*, but not necessary, for generating a t error correcting code is:

$$2^k > \sum_{i=0}^{2t-1} C_{n-1}^i, \quad (5.45)$$

known as Varshamov-Gilbert boundary.

For one error correction, the two boundaries lead to the same equation, which is the necessary and sufficient condition for generating a one error correcting code:

$$2^k \geq 1+n \quad (5.46)$$

Remark

Demonstrations of the previous relations are given in [2] and [43],[50].

5.7.4 Perfect and almost Perfect Codes

The *perfect (tightly packed/lossless) codes* are the ones that fulfil the following equation [2], [50]:

$$\sum_{i=1}^t C_n^i = 2^k - 1 \quad (5.47)$$

Perfect codes can correct exactly t errors, but any particular configuration with more than t errors results in even more errors.

There are only few perfect codes; so far known are: Hamming codes with $n = 2^k - 1$, binary code with repetition for even n and two Golay codes [2], [50], [54].

The *almost perfect codes* correct all the t errors and some $N \leq C_n^i$ other combinations of $t + 1$ errors, under the assumption that (for binary codes):

$$\sum_{i=1}^t C_n^i + N = 2^k - 1 \quad (5.48)$$

The perfect and almost perfect codes have a maximum probability of the correct reception, when transmitting over symmetrical channel with independent errors.

5.7.5 Detection and Correction Capacity: Decoded BER

Let us consider the error detecting linear block code $C(n,m)$. The total number of errors N_{te} is:

$$N_{te} = 2^n - 1 \quad (5.49)$$

and the number of undetectable errors is:

$$N_{ue} = 2^m - 1 \quad (5.50)$$

Detection capacity, according to (5.11) will be:

$$C_d = \frac{N_{de}}{N_{te}} = \frac{N_{te} - N_{ue}}{N_{te}} = 1 - \frac{N_{ue}}{N_{te}} = 1 - \frac{2^m - 1}{2^n - 1} \cong 1 - \frac{2^m}{2^n} = 1 - 2^{-k} \quad (5.51)$$

Similarly, using the definition (5.12) we can calculate the correction capacity C_c , with the remark that we must take into account if the code is a perfect or an almost perfect one.

In the case of an error detecting code $C(n,m)$ used in a BSC, the undetectable error probability P_u is calculated considering that the error is undetectable if another non-zero codeword is obtained. It follows that [28]:

$$P_u = \sum_{i=1}^n A_i (1-p)^{n-i} \quad (5.52)$$

where by A_i we denoted the codewords number with the weight i belonging to the set C . The numbers A_0, \dots, A_n form the *weights distribution of C* .

Remark

- For a code $C(n,m)$ having the code distance d we get $A_0 = \dots = A_{d-1} = 0$.

In theory we can compute weight distribution for any linear code (n,m) by evaluating the 2^m codewords. However, for high n and m the calculus becomes

practically impossible; except for some short linear codes the weight distribution is not yet determined, so P_u is unknown.

Even so, it is possible to determine the *upper bound of the error probability after decoding* [28] as follows:

$$P_u \leq 2^{-k} \sum_{i=1}^n C_n^i p^i (1-p)^{n-i} = 2^{-k} [1 - (1-p)^n] \cong 2^{-k} \quad (5.53)$$

since $[1 - (1-p)^n] \leq 1$.

Equation (5.53) shows that there exist linear codes (n, m) for which P_u decreases exponentially with the number of control bits k .

Remark

Although if the number of linear block codes is extremely great, only a restrained category of codes proved to have P_u satisfying the upper bound 2^{-k} (the Hamming codes, for example).

A t error correcting block code $C(n, m)$, excluding perfect codes, can correct numerous combinations of $t+1$ errors and even more. The number of correctable combinations is $2^k - 1$. When the transmission is performed over a BSC, a block decoding error probability has an upper bound given by [28]:

$$P \leq \sum_{i=t+1}^n C_n^i p^i (1-p)^{n-i} \quad (5.54)$$

In the previous formula, P represents the block erroneous decoding probability. For perfect codes, equation (5.54) is fulfilled at limit. In order to determine the after decoding bit error rate (p_d) we start with the definition of p_d :

$$\begin{aligned} p_d &= \frac{\text{no. of erroneous bits after decoding}}{\text{total no. of transmitted bits}} = \\ &= \frac{\text{no. of erroneous blocks} \times \text{no. of erroneous bits per block}}{\text{total no. of blocks} \times n} \quad (5.55) \\ &= \frac{\text{no. of erroneous bits per block}}{n} \times p = \frac{\gamma}{n} \times p \end{aligned}$$

where γ is the number of erroneous bits from an erroneous block of length n . The value of γ depends on channel errors statistics and also on the code. For independent errors, and one error correcting codes we have $\gamma = 2 \div 3$ while for $t = 2$ errors correcting codes, $\gamma = 3 \div 4$ [47].

5.7.6 Relations between the Columns of H Matrix for Error Detection and Correction

We have shown in 5.7.2 that in error detection the syndrome S must be non zero. Let us now analyze the implications of this condition on the control matrix structure. Consider an erroneous word \mathbf{e}_i with t errors:

$$\mathbf{e}_i = [0 \dots e_{i_1} \dots e_{i_t} \dots 0] \quad (5.56)$$

where $e_{i,j} = 1$ in the positions on which errors occur.

In this case equation (5.48) can be written as follows:

$$\mathbf{S} = [\mathbf{h}_1 \quad \mathbf{h}_2 \quad \dots \quad \mathbf{h}_n] \cdot \begin{bmatrix} 0 \\ \dots \\ \mathbf{e}_i \\ \dots \\ \mathbf{e}_{i_t} \\ \dots \\ 0 \end{bmatrix} = \mathbf{h}_{i_1} \oplus \mathbf{h}_{i_2} \oplus \dots \oplus \mathbf{h}_{i_t} \neq \mathbf{0}, \forall i_k = \overline{1, n} \quad (5.57)$$

This means that for t errors detection, the modulo two summation of any t columns of the \mathbf{H} matrix must be non zero.

In particular, for detecting one error ($t = 1$), all the control matrix columns must be non zero; however, the columns may be equal.

For correcting maximum t errors we must have distinct syndromes for all the t errors possible combinations; in this case equation (5.48) becomes:

$$\mathbf{S}_i = \mathbf{H}\mathbf{e}_i^T \neq \mathbf{S}_j = \mathbf{H}\mathbf{e}_j^T \quad (5.58)$$

where \mathbf{e}_i and \mathbf{e}_j are two distinct combinations of t errors.

Expressing \mathbf{H} by its columns, we obtain:

$$[\mathbf{h}_1 \quad \mathbf{h}_2 \quad \dots \quad \mathbf{h}_n] \begin{bmatrix} 0 \\ \dots \\ e_{i_1} \\ \dots \\ e_{i_2} \\ \dots \\ 0 \end{bmatrix} \neq [\mathbf{h}_1 \quad \mathbf{h}_2 \quad \dots \quad \mathbf{h}_n] \begin{bmatrix} 0 \\ \dots \\ e_{j_1} \\ \dots \\ e_{j_t} \\ \dots \\ 0 \end{bmatrix}$$

or:

$$\mathbf{h}_{i_1} \oplus \mathbf{h}_{i_2} \oplus \dots \oplus \mathbf{h}_{i_t} \neq \mathbf{h}_{j_1} \oplus \mathbf{h}_{j_2} \oplus \dots \oplus \mathbf{h}_{j_t}, \forall i_k, j_k = \overline{1, n} \quad (5.58.a)$$

Adding on both sides columns $\mathbf{h}_{j_1}, \mathbf{h}_{j_2}, \dots, \mathbf{h}_{j_t}$ and reordering the columns we obtain:

$$\mathbf{h}_{i_1} \oplus \mathbf{h}_{i_2} \oplus \dots \oplus \mathbf{h}_{i_{2t}} \neq 0, \forall i_k = \overline{1, n} \quad (5.58.b)$$

Equation (5.58.b) shows that for t errors correction the modulo two summation of any $2t$ columns of the \mathbf{H} matrix must be non zero.

For the particular case of one error correcting code, all the columns of the control matrix must be non zero and distinct from each other.

5.7.7 Standard Array and Syndrome Decoding

For linear block codes $C(n,m)$, the codewords number is 2^m . No matter what word is transmitted, at receiver we can obtain any of the 2^n possible combinations of V_n . Any decoding algorithm should divide the set V_n into distinct subsets D_i , each of them containing only one code word v_i , $1 \leq i \leq 2^m$. If the received word r is in the subset D_i it will be decoded v_i .

This decoding method, which consists in partitioning the received words set V_n into distinct subsets D_i that contain one codeword, is known as the *standard array*. The standard array represents the operation of partitioning the group V_n after the subgroup C by drawing a table with all the 2^n received words. The table is filled as follows:

- first row (class) will contain all the 2^m codewords starting from the all zero word placed in the first left column
- each row forms a class; the first element from each class placed in the first column of the table is called the *coset leader*
- from the remaining $2^m - 2^n$ combinations (not used in the first class) $2^k - 1$ elements are chosen to form the coset leaders of the next classes; we choose these elements (denoted with e_j , $j=1,2^k$) in such a way that the decoding error probability is minimized, therefore we chose the error combinations with the highest probability of occurrence.

This algorithm is illustrated in table 5.1.

Table 5.1 Standard array for a $C(n,m)$ code.

Coset leader $e_1 = v_1 = (0,0,\dots,0)$	D_2 v_2	D_3 v_3	...	D_i v_i	...	D_{2^m} v_{2^m}
e_2	$e_2 \oplus v_2$	$e_2 \oplus v_3$...	$e_2 \oplus v_i$...	$e_2 \oplus v_{2^m}$
...
e_j	$e_j \oplus v_2$	$e_j \oplus v_3$...	$e_j \oplus v_i$...	$e_j \oplus v_{2^m}$
...
e_{2^k}	$e_{2^k} \oplus v_2$	$e_{2^k} \oplus v_3$...	$e_{2^k} \oplus v_i$...	$e_{2^k} \oplus v_{2^m}$

For a BSC the error words with minimum weight have a maximum probability of occurrence, so the coset leaders of each class will be chosen from the n -dimensional combinations which are not codewords and have a minimum weight

(will be denoted with $\mathbf{e}_2, \dots, \mathbf{e}_{2^m}$). Each class, i.e. the elements of each row will be obtained from $\mathbf{v}_i \oplus \mathbf{e}_j$, $i=1, 2^m$.

Both the table and the decoding are based on the minimum distance algorithm MLD. Decoding errors occur only if the error combinations are not in the table of the coset leaders.

For a BSC the probability of a decoding error (at word level) is [28]:

$$P = 1 - \sum_{i=0}^n \alpha_i p^i (1-p)^{n-i} \quad (5.59)$$

where the numbers $\alpha_0, \alpha_1, \dots, \alpha_n$ form the weight distribution of the main elements (from the first left column) and p is the BER of a BSC.

Example 5.4

Let us draw the table corresponding to the standard array for the code $C(5,3)$ analysed in Example 5.3. The first class (first line) is made up by eight codewords, the first being the one with all zero elements. Next we select $2^k = 2^{5-3} = 2^2 = 4$ possible error combinations with minimum weight. Having given $\mathbf{e}_1=(0\ 0\ 0\ 0\ 0)$, it results that we must choose another three combinations with unit weight: $\mathbf{e}_2=(1\ 0\ 0\ 0\ 0)$, $\mathbf{e}_3=(0\ 1\ 0\ 0\ 0)$ and $\mathbf{e}_4=(0\ 0\ 1\ 0\ 0)$. The table corresponding to the standard array is given in Table 5.2

Table 5.2 Standard array for $C(5,3)$

00000	10010	01011	00101	11001	10111	01110	11100
10000	00010	11011	10101	01001	00111	11110	01100
01000	11010	00011	01101	10001	11111	00110	10100
00100	10110	01111	00001	11101	10011	01010	11000

Let us consider the reception of sequence $\mathbf{r}=01101$. Looking in the table we estimate \mathbf{r} as $\mathbf{v}_4=00101$, obtained by the erronation of the second left sided bit.

One may notice that this code can correct only some combinations from the total possible one error combinations; this was expected as the code distance of this code is $d = 2$, so it does not fulfil the necessary condition to correct at least one error ($d = 3$ for $t = 1$).

Syndrome decoding (using lookup table)

From the table containing the standard array for a linear code $C(n,m)$ one must notice that all the 2^m elements from the same class have the same syndrome [28], so there are 2^k different syndromes corresponding to their classes. Therefore a bi-univocal correspondence exists between the coset leader of a certain class and its syndrome; it results that a lookup table may be determined using the 2^k leaders (corresponding to the correctable error combinations) and their syndromes. This

table can be memorized at receiver. In this case the decoding is performed as follows (table 5.3):

- compute the syndrome corresponding to \mathbf{r} :

$$\mathbf{S} = \mathbf{H}\mathbf{r}^T$$

- identify the coset leader \mathbf{e}_j for which we have the same syndrome with the one determined for \mathbf{r} ; decide that \mathbf{r} comes being erroneous by \mathbf{e}_j
- decode \mathbf{r} as:

$$\mathbf{v} = \mathbf{r} + \mathbf{e}_j$$

Table 5.3 Syndrome decoding for a linear code $C(n,m)$

\mathbf{S}_i	\mathbf{e}_i
\mathbf{S}_1	\mathbf{e}_1
\mathbf{S}_2	\mathbf{e}_2
...	...
\mathbf{S}_2^m	\mathbf{e}_2^m

Example 5.5

Use the syndrome decoding algorithm to decode the linear code $C(5,3)$ from Example 5.3.

In order to calculate the syndromes, according to (5.47), we need to generate \mathbf{H} . The generating matrix has already been determined as:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix} = [\mathbf{I}_3 \ \mathbf{P}]$$

The matrix \mathbf{G} is already in a canonical form; according to (5.36), \mathbf{H} is:

$$\mathbf{H} = \left[\begin{array}{ccc|cc} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ \hline & \mathbf{P}^T & & \mathbf{I}_2 \end{array} \right]$$

The syndromes corresponding to the three combinations of correctable errors $\mathbf{e}_2 = (1\ 0\ 0\ 0\ 0)$, $\mathbf{e}_3 = (0\ 1\ 0\ 0\ 0)$ and $\mathbf{e}_4 = (0\ 0\ 1\ 0\ 0)$ are:

$$\mathbf{S}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \mathbf{S}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{S}_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

The syndrome decoding table is:

S_i	e_i
00	00000
10	10000
11	01000
01	00100

5.7.8 Comparison between Linear Error Detecting and Correcting Block Codes

An estimation of code efficiency is performed from the user point of view, the user giving the fidelity degree (either by the maximum error probability or by the minimum value of the signal/noise ratio SNR) taking into consideration a series of variables, such as: cost, transmission speed, complexity etc.

The detection/correction capacity C_d/C_c defined in 5.5 as well as the BER equations defined in the previous paragraphs do not take into consideration the transmission parameters: power of the signal P_s , channel noise P_N , information bit rate at the user (\dot{D}_i).

An efficient comparison from a practical point of view must take into account all these parameters.

The hypothesis in which we compare the error detecting and correcting codes are:

- same power at transmission: $P_s = ct$
- a transmission channel with the spectral density power of noise N_0 known, assumed to be a BSC with $p \ll 1$
- information bit rate at the receiver equal to the one of the uncoded source: $\dot{D}_i = ct$
- receiving and transmission are assumed independent

Comparison of error correcting codes

Let us consider two error correcting codes that correct t_1 , respectively t_2 errors. Considering perfect codes, equation (5.54) is valid at the limit (the most disadvantageous case): $C_1(n_1, m_1, t_1)$ and $C_2(n_2, m_2, t_2)$.

One question raises: which one of these two codes is better? It is not always true that the most redundant code is also the best. We have seen in 5.1 that by redundancy increasing we obtain the receiver bandwidth increasing and subsequently power noise extension, therefore the error probability increasing.

The criterion taken into consideration when estimating which one of the two codes is better (also taking into account the transmission parameters) is the correct

decision probability for the same quantity of information transmitted over the channel. In this case:

$$N_1 m_1 = N_2 m_2 \tag{5.60}$$

where N_1 , respectively N_2 , are the number of codewords transmitted through C_1 , C_2 codes.

The most efficient code will be the one for which the correct transmission probability is greater:

$$(1 - P_1)^{N_1} \underset{C_2}{\overset{C_1}{>}} (1 - P_2)^{N_2} \tag{5.61}$$

where P_1 and P_2 are the erroneous decoding probabilities for codes C_1 and C_2 and are determined with (5.54) at the limit:

$$P_1 = \sum_{i=t_1+1}^{n_1} C_{n_1}^i p^i (1-p)^{n_1-i} \cong C_{n_1}^{t_1+1} p^{t_1+1}$$

$$P_2 = \sum_{i=t_2+1}^{n_2} C_{n_2}^i p^i (1-p)^{n_2-i} \cong C_{n_2}^{t_2+1} p^{t_2+1} \tag{5.62}$$

The approximations in (5.62) stand for a BSC with $p \ll 1$ case.

The fact that the BER in the channel are different for the two codes is due to the different bit rates requiring different bandwidths; according to equations (2.80) and (5.6) we have:

$$B_1 = \frac{1}{2} \dot{D}_{c_1} = \frac{1}{2} \frac{n_1}{m_1} \dot{D}_i$$

$$B_2 = \frac{1}{2} \dot{D}_{c_2} = \frac{1}{2} \frac{n_2}{m_2} \dot{D}_i \tag{5.63}$$

Relation (5.61) becomes (when $P_{1,2} \ll 1$):

$$1 - N_1 P_1 \underset{C_2}{\overset{C_1}{>}} 1 - N_2 P_2$$

or

$$P_1 \underset{C_2}{\overset{C_1}{>}} \frac{m_1}{m_2} P_2 \tag{5.64}$$

Replacing in (5.64) P_1 and P_2 with the expressions from (5.62) we have:

$$m_2 C_{n_1}^{t_1+1} p^{t_1+1} \begin{matrix} > \\ < \end{matrix} \begin{matrix} C_1 \\ C_2 \end{matrix} m_1 C_{n_2}^{t_2+1} p^{t_2+1} \tag{5.65}$$

The BER on the channel depends on the source bit rate ($\dot{D}_{c_1}, \dot{D}_{c_2}$) and on the modulation and decision systems used at receiver [15], [28] [Appendix C].

Be p given by the relation (C.40) from Appendix C:

$$p = Q(\sqrt{\xi}) \tag{5.66}$$

where ξ is the SNR:

$$\xi = \frac{P_S}{P_N} = \frac{P_S}{N_0 B} \tag{5.67}$$

and Q is the coerror function [Appendix C].

$$Q(y_1) = \int_{y_1}^{\infty} \frac{1}{\sqrt{2p}} e^{-\frac{1}{2}y^2} dy \tag{5.68}$$

Under these circumstances equation (5.56) becomes:

$$m_2 C_{n_1}^{t_1+1} \left[Q\left(\sqrt{2 \frac{P_S}{N_0 \dot{D}_i} \frac{m_1}{n_1}} \right) \right] \begin{matrix} > \\ < \end{matrix} \begin{matrix} C_1 \\ C_2 \end{matrix} m_1 C_{n_2}^{t_2+1} \left[Q\left(\sqrt{2 \frac{P_S}{N_0 \dot{D}_i} \frac{m_2}{n_2}} \right) \right]^{t_2+1} \tag{5.69}$$

Comparison between error detecting codes

Judging as in error correcting codes case, we use (5.69) replacing $C_{n_1}^{t_1+1}, C_{n_2}^{t_2+1}$ with N_1 and N_2 - the number of different sequences that can be received with $t_1 + 1$ and, respectively $t_2 + 1$ undetectable errors. It is obvious that all sequences with t_1, t_2 error are detected by the two codes.

Coding gain G_c

In practice, to estimate the code error protection efficiency, is used the parameter *coding gain* G_c , defined as the difference between the SNR necessary for an uncoded transmission ξ_{nc} and the one corresponding to the encoded transmission ξ_c for the same BER at the user (Fig. 5.7).

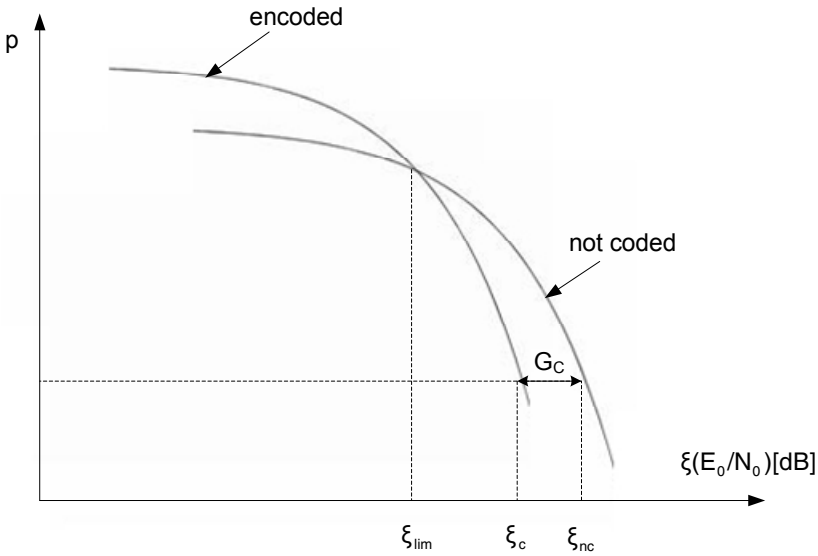


Fig. 5.7 Coding gain representation

$$G_c|_{p=ct} = \xi_{nc} - \xi_c \tag{5.70}$$

From the graphic of G_c one may notice that there is a certain limit value of the SNR (ξ_{lim}) below which the uncoded transmission BER is less than for the coded one. One intuitive explanation is that an error protection code is redundant, meaning that for $\dot{D}_i = ct$, \dot{D}_c increases; it follows a bandwidth expansion and, subsequently a noise increase, therefore a decrease for ξ , having as consequence the increase of the error probability (BER).

Concluding, to have a BER imposed by the application, at the user, we must decide upon one of the followings:

- increasing the SNR (increasing the power at transmission, using efficient modulation systems)
- using error protection codes, the price paid being a bandwidth increasing; generally speaking, this method proves to be more efficient in many applications.

5.7.9 Hamming Group Codes

Hamming codes are the first error correcting codes proposed after Shannon second theorem, by R. Hamming in 1950.

One error correcting (t=1) Hamming group codes (perfect)

The characteristics of this code are:

- code length can be determined by:

$$n = 2^k - 1 \quad (5.71)$$

This equation is, in fact, a perfect code existence condition (5.43) for $t = 1$.

- the code is separable, and has the structure:

$$\mathbf{v} = [c_1 \ c_2 \ a_3 \ c_4 \ a_5 \ a_6 \ a_7 \ c_8 \ a_9 \dots a_n] \quad (5.72)$$

where by a_i we denoted the information symbols and by c_i the control (parity check) symbols.

- control symbols are placed on positions

$$2^i, \quad i = \overline{0, k-1} \quad (5.73)$$

- control matrix \mathbf{H} is:

$$\mathbf{H}_{[k \times n]} = [\mathbf{h}_1 \ \dots \ \mathbf{h}_i \ \dots \ \mathbf{h}_n] \quad (5.74)$$

where each column \mathbf{h}_i expresses in binary natural code (BN) its position with the less significant bit (LSB) on the k -th line.

From the control matrix structure one may notice that all the \mathbf{h}_i columns are distinct, therefore condition (5.58.b) is fulfilled in the case of one error correcting ($t = 1$) codes.

The encoding relations are determined using (5.38):

$$\mathbf{H}\mathbf{v}^T = 0$$

It follows that the control symbols are expressed as a linear combination of information symbols:

$$c_i = f_i(a_i)$$

Remark

Relationship (5.38) expresses that even parity is checked ($=0$).

Be an error on the i -th position:

$$\mathbf{e} = [0 \ \dots \ e_i \ \dots \ 0], \quad e_i = 1$$

The syndrome is determined using (5.48):

$$\mathbf{S} = [\mathbf{h}_1 \ \dots \ \mathbf{h}_n] \begin{bmatrix} 0 \\ \dots \\ e_i \\ \dots \\ 0 \end{bmatrix} = \mathbf{h}_i \quad (5.75)$$

When only one error occurs, the syndrome indicates the error position in binary; by a binary-decimal conversion we may determine from \mathbf{S} the error word \mathbf{e} .

Error correction is easy:

$$\mathbf{v} = \mathbf{r} \oplus \mathbf{e} = (\mathbf{v} \oplus \mathbf{e}) \oplus \mathbf{e} = \mathbf{v} \quad (5.76)$$

Let us now see what happens when two errors occur ($t = 2$) on the i -th and j -th positions:

$$\mathbf{e} = [0 \quad \dots \quad e_i \dots e_j \quad \dots \quad 0]$$

In this case the error syndrome \mathbf{S} is:

$$\mathbf{S} = [\mathbf{h}_1 \quad \dots \quad \mathbf{h}_n] \begin{bmatrix} 0 \\ \dots \\ e_i \\ \dots \\ e_j \\ \dots \\ 0 \end{bmatrix} = \mathbf{h}_i \oplus \mathbf{h}_j = \mathbf{h}_k \quad (5.77)$$

It may be easily noticed that, if two errors occur, the code introduces a third error on the k -th position.

For this code, a block error probability is determined with (5.54) for $t = 1$:

$$P = \sum_{i=2}^n C_n^i p^i (1-p)^{n-i} \cong C_n^2 p^2 (1-p)^{n-2} \cong \frac{n(n-1)}{2} \cdot p^2 \quad (5.78)$$

We remind to the reader that (5.78) has been determined under the assumption that the errors are independent and the channel is a BSC one with p as BER.

The BER after Hamming decoding is obtained with (5.55): $p_d = \frac{\gamma}{n} \cdot P$.

We have seen that for the perfect Hamming code, if two errors ($t = 2$) occur, another one is introduced. Therefore, the most probable number of errors is $\gamma = 3$, such that:

$$p_d = \frac{3}{n} \cdot P = \frac{3}{n} \cdot \frac{n(n-1)}{2} \cdot p^2 \cong \frac{3}{2} \cdot (n-1) \cdot p^2 \quad (5.79)$$

Modified Hamming Codes

As we have already seen, the one error correcting Hamming code (the perfect code) introduces supplementary errors when overloaded with more errors than its correcting capacity ($t > 1$). In order to eliminate this disadvantage and to make it more useful in practical applications, this code has been modified by increasing the minimum distance from $d = 3$ to $d = 4$, allowing one error correction and double error detection. This modification is possible either extending or shortening the initial code. In what follows we shall briefly present both cases.

Extended Hamming Codes

Increasing the code distance from 3 to 4 is performed adding a supplementary control symbol (parity control symbol) c_0 . The codeword structure becomes:

$$\mathbf{v}^* = [c_0 \ c_1 \ c_2 \ a_3 \ c_4 \ a_5 \ a_6 \ a_7 \ c_8 \ a_9 \ \dots a_n] \quad (5.80)$$

The control matrix will be:

$$\mathbf{H}^* = \begin{bmatrix} \mathbf{0} & \mathbf{H} \\ \mathbf{1} & \mathbf{1} \end{bmatrix} \quad (5.81)$$

where \mathbf{H} is the Hamming perfect code matrix given by (5.74).

In this case the syndrome is:

$$\mathbf{S} = \mathbf{H}^* \mathbf{v}^{*T} = \begin{bmatrix} \mathbf{0} & \mathbf{H} \\ \mathbf{1} & \mathbf{1} \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ \cdot \\ a_n \end{bmatrix} = \begin{bmatrix} \mathbf{S}_1 \\ \mathbf{S}_2 \end{bmatrix} \quad (5.82)$$

where \mathbf{S}_1 keeps its significance from the Hamming perfect code and \mathbf{S}_2 is a binary symbol, “0” or “1”; using \mathbf{S}_2 we may detect even errors ($\mathbf{S}_2 = 0$).

We may have one of the following cases:

$$\left. \begin{array}{l} \mathbf{S}_1 = \mathbf{0} \\ \mathbf{S}_2 = 0 \end{array} \right\} \Rightarrow \text{no errors or the errors are not detectable} \quad (5.83.a)$$

$$\left. \begin{array}{l} \mathbf{S}_1 \neq \mathbf{0} \\ \mathbf{S}_2 = 1 \end{array} \right\} \Rightarrow \text{one correctable error in the interval } \overline{1 \div n} \quad (5.83.b)$$

$$\left. \begin{array}{l} \mathbf{S}_1 = \mathbf{0} \\ \mathbf{S}_2 = 1 \end{array} \right\} \Rightarrow c_0 \text{ symbol is erroneous} \quad (5.83.c)$$

$$\left. \begin{array}{l} \mathbf{S}_1 \neq \mathbf{0} \\ \mathbf{S}_2 = 0 \end{array} \right\} \Rightarrow \text{two errors are detected} \quad (5.83.d)$$

The code distance is $d = 4$ and it corresponds to one error correction and two errors detection: $d \geq t + e + 1$ ($4 = 1 + 2 + 1$).

Example 5.6

The odd parity code $H(8,4)$ used in teletext systems is given by the control matrix:

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

The structure of a codeword is:

$$\mathbf{v}^* = [c_0 c_1 a_2 c_3 a_4 a_5 a_6 c_7]$$

Encoding relation, taking into account the odd parity, becomes:

$$\mathbf{H}^* \mathbf{v}^{*\text{T}} = \mathbf{1}$$

Encoding relations are determined from:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} c_0 \\ c_1 \\ a_2 \\ c_3 \\ a_4 \\ a_5 \\ a_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

It follows that:

$$c_1 = a_5 \oplus a_4 \oplus a_2 \oplus 1$$

$$c_3 = a_6 \oplus a_4 \oplus a_2 \oplus 1$$

$$c_7 = a_6 \oplus a_5 \oplus a_4 \oplus 1$$

$$c_0 = c_1 \oplus a_2 \oplus c_3 \oplus a_4 \oplus a_5 \oplus a_6 \oplus c_7 \oplus 1$$

The syndrome is determined from the equation:

$$\mathbf{S} = \mathbf{H}^* \mathbf{r}^{*\text{T}} \oplus \mathbf{1}$$

Shortened Hamming Codes

The same increase of the code distance from 3 to 4 can be obtained shortening a perfect Hamming code. This can be obtained eliminating from the perfect code control matrix \mathbf{H} all columns having an even number of ones, that is, the columns corresponding to the information positions. The new obtained code corrects one error and detects two. If only one error occurs during the transmission, the syndrome is not $\mathbf{0}$ and it contains an odd number of ones. Decoding is performed as follows [28]:

- $\mathbf{S} = \mathbf{0} \Rightarrow$ no errors or undetectable errors
- $\mathbf{S} \neq \mathbf{0}$ and containing an odd number of ones \Rightarrow one error; correction is performed adding the syndrome corresponding to the error word to the received word (erroneous)
- $\mathbf{S} \neq \mathbf{0}$ and containing an even number of ones \Rightarrow two errors are detected.

Shortened Hamming codes have important applications in computers internal memory protection. These codes have been studied and proposed for applications in the memory protection field by Hsiao [23]; he also provided an algorithm for obtaining the optimal control matrix \mathbf{H}_0 with the following properties:

- each column has an odd number of “1”s
- the total number of “1”s is minimum
- the number of “1”s from each line in \mathbf{H}_0 must be equal or as close as possible to the average number of ones per line.

Fulfilling these three conditions, an optimal \mathbf{H}_0 matrix with $d \geq 4$ and a minimum hardware is obtained.

When dealing with applications in computers field, these codes are encoded and decoded in parallel, taking into account that speed is the most critical problem of the on-line processing.

Example 5.7 [47]

For 16 bits memory protection, Hsiao introduced an optimal one error correcting and two error detecting ($d = 4$) code $H_0(22,16)$ obtained as follows:

- start from the perfect $H(63,57)$ code with $n = 2^6 - 1$.
- shorten the code eliminating the 31 columns that contain an even number of ones, and obtain the code: $H(63 - 31, 57 - 31) = H_s(32, 26)$
- for 16 bits memory ($m = 16$), eliminate ten more columns, complying the optimal conditions for obtaining a minimum hardware [23].

Finally the obtained shortened Hamming code is $H_0(22,16)$:

$$\mathbf{H}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Figure 5.8 shows the way to use this code for 16 bits memory protection.

In a memory writing cycle, 6 parity bits are generated and stored in a check memory.

In a read cycle, new parity bits are generated from the received data and compared to the ones found in the control memory. If errors are detected, the $H(22, 16)$ block disconnects the central unit (CU).

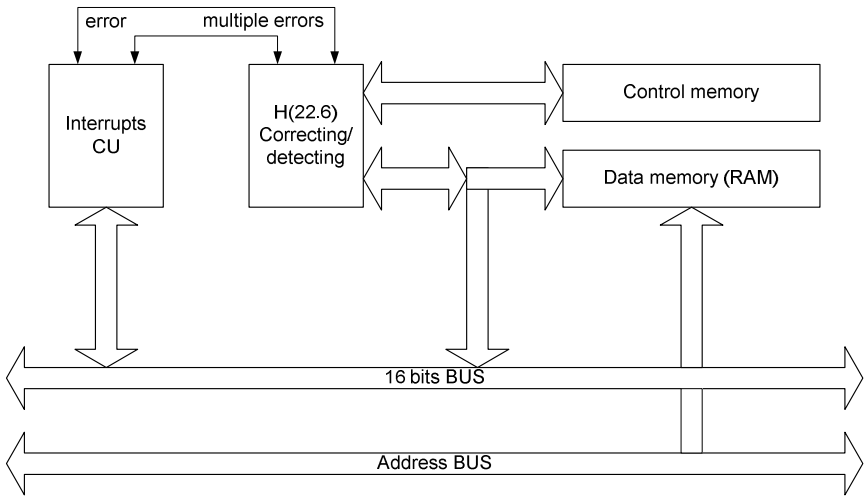


Fig. 5.8 Shortened Hamming code for 16 bits memory protection

The data format for the H(22,16) code is:

Data		control bits
Byte 1	Byte 0	6

Similarly, for 32 bits memory protection the shortened Hamming code H(39,32) [23] is determined.

Example 5.8

Let us consider the error protection of a digital transmission performed with a one error correcting Hamming code with $m = 4$.

- Dimension the code, determine the encoding relations using **H** and **G** matrices and design the encoding and decoding block schemes for a maximal processing speed; which is the code distance and how can it be determined?
- Is $\mathbf{r} = [1101101]$ a codeword? If not, determine the corresponding correct word assuming only one error in transmission (LSB is the first value from left in the word \mathbf{r}).
- How much is the BER after decoding if the transmission is performed on a BSC with $p = 10^{-2}$?
- Repeat point b) assuming standard array and syndrome-based decoding; discussion.
- Analyze the possibility of increasing the code distance from 3 to 4; discussion.

Solution

a) For $m = 4$, determine k using equation (5.71):

$$n = 2^k - 1 = m + k \Rightarrow k = 3$$

from where it results the H(7,4) code.

According to (5.74) the control matrix has the following expression:

$$\mathbf{H} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

In order to determine the correspondent generating matrix, it is necessary to determine a canonical form of \mathbf{H} matrix and then the correspondent canonical form for \mathbf{G} ; then, by rearranging the columns, one can obtain \mathbf{G} matrix.

$$\mathbf{H}' = \begin{bmatrix} 3 & 5 & 6 & 7 & 4 & 2 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \Rightarrow \mathbf{G}' = \begin{bmatrix} 3 & 5 & 6 & 7 & 4 & 2 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$\mathbf{P}^T \qquad \mathbf{I}_3 \qquad \mathbf{I}_4 \qquad \mathbf{P}$

$$\mathbf{G} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Remark

The computation accuracy is checked if the two matrices are orthogonal (relation (5.37)). We advice the reader to check this result.

The encoding relations (5.38) and (5.33) are giving the control symbols as functions of the information symbols.

According to (5.72), the codeword structure is the following:

$$\mathbf{v} = [c_1 c_2 a_3 c_4 a_5 a_6 a_7]$$

The encoding relations using the control matrix (5.38) are the followings:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ a_3 \\ c_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = 0 \Rightarrow \begin{cases} c_4 = a_5 \oplus a_6 \oplus a_7 \\ c_2 = a_3 \oplus a_6 \oplus a_7 \\ c_1 = a_3 \oplus a_5 \oplus a_7 \end{cases}$$

The encoding relations, using the generating matrix, (5.33), obviously lead to the same result:

$$[a_3 \ a_5 \ a_6 \ a_7] \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} = [c_1 c_2 a_3 c_4 a_5 a_6 a_7]$$

The code distance can be determined in more ways:

- writing the codewords set C and determining their weights using equation (5.16); it is found $d = w_{\min} = 3$; we invite the reader to make this calculus, as an exercise.
- one may notice that all the columns of \mathbf{H} matrix are distinct, therefore, according to (5.58.b) the code can correct one error; it follows, according to (5.25): $d = 2 \cdot 1 + 1 = 3$

The decoding equation (5.47) becomes:

$\mathbf{S} = \mathbf{H}\mathbf{r}^T$; from which it follows:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \\ r_7 \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} \Rightarrow \begin{cases} r_4 \oplus r_5 \oplus r_6 \oplus r_7 = s_1 \\ r_2 \oplus r_3 \oplus r_6 \oplus r_7 = s_2 \\ r_1 \oplus r_3 \oplus r_5 \oplus r_7 = s_3 \end{cases}$$

The previously determined encoding and decoding equations lead to the block schemes of the encoding/decoding units (Fig 5.9).

According to (5.76) the error correction consists in:

$$\mathbf{v} = \mathbf{r} \oplus \mathbf{e}$$

where \mathbf{e} is determined from \mathbf{S} by a binary-natural to decimal conversion.

For Hamming code, the correction is performed when \mathbf{r} has been completely received (on a fraction from the duration of the n -th symbol), the seven modulo two adders being validated and allowing the writing of the correct word in the memory register (MR): $\mathbf{v} = \mathbf{r} \oplus \mathbf{e}$.

b) To check whether \mathbf{r} is or not a codeword, check the equation (5.47):

$$\mathbf{H}\mathbf{r}^T = \mathbf{S}$$

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \mathbf{h}_5$$

Decoding \mathbf{h}_5 one can determine the error word:

$$\mathbf{e} = [00000100]$$

Correction consists in:

$$\mathbf{v} = \mathbf{r} \oplus \mathbf{e} = [1101101] \oplus [0000100] = [1101001]$$

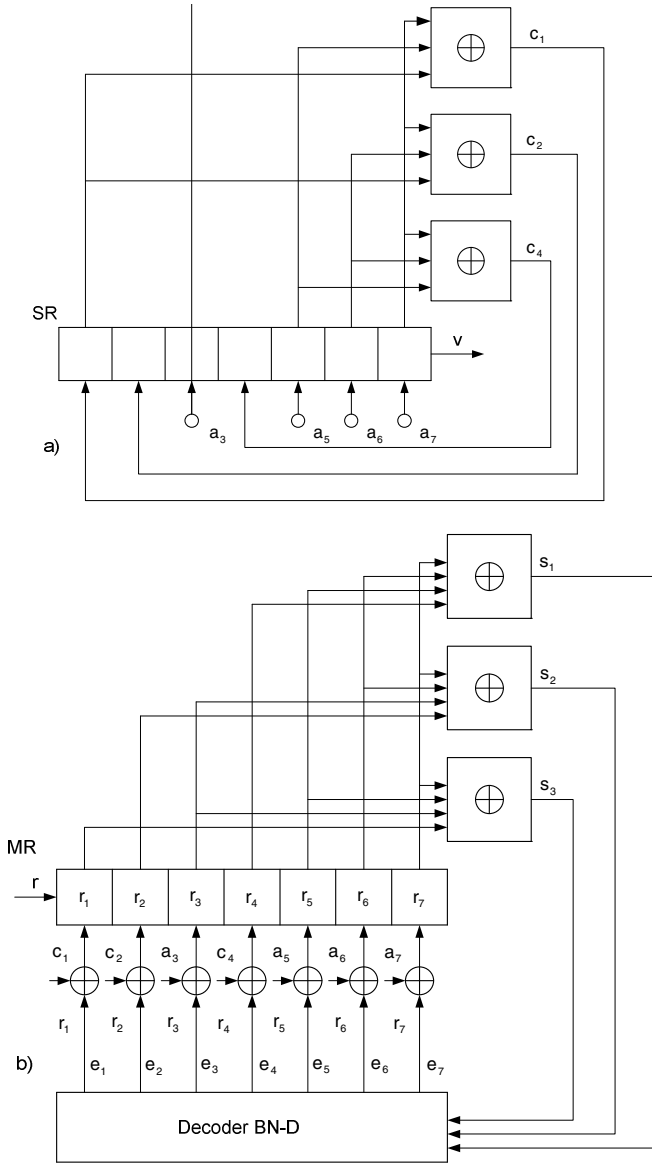


Fig. 5.9 Hamming (7,4) code block scheme: a) encoding unit; b) decoding unit

c) p_d is determined with equation (5.79):

$$p_d \cong \frac{3}{2}(n-1)p^2 = 1,5 \cdot 6p^2 = 9 \cdot 10^{-4} \cong 10^{-3},$$

so it decreased approximately 10 times, compared to the situation of not using the error correction code.

d) For our code, the codewords set is formed by $2^m = 16$ combinations of 7 bits.

[0 0 0 0 0 0 0], [1 1 0 1 0 0 1], [0 1 0 1 0 1 0], [1 0 0 0 0 1 1],
 [1 0 0 1 1 0 0], [0 1 0 0 1 0 1], [1 1 0 0 1 1 0], [0 0 0 1 1 1 1],
 [1 1 1 0 0 0 0], [0 0 1 1 0 0 1], [1 0 1 1 0 1 0], [0 1 1 0 0 1 1],
 [0 1 1 1 1 0 0], [1 0 1 0 1 0 1], [0 0 1 0 1 1 0], [1 1 1 1 1 1 1].

In order to form the standard array we choose from the rest of $2^n - 2^m = 2^7 - 2^4 = 112$ combinations on 7 bits, a number of $2^k = 8$ combinations of minimum weight except the combination with all zero elements; we shall choose the 7 combinations with unitary weight (see Table 5.4).

The sequence $\mathbf{r} = [1101101]$ is identified at the intersection of line 6 with column 2, so the decoded word is 1101001.

Remark

Due to the fact that the table contains all possible combinations corresponding to one error, any singular error can be corrected.

For syndrome-based decoding, we fill Table 5.5.

Table 5.5 Syndrome-decoding table for Hamming (7,4) code.

\mathbf{S}^T	\mathbf{e}
0 0 0	0 0 0 0 0 0 0
0 0 1	1 0 0 0 0 0 0
0 1 0	0 1 0 0 0 0 0
0 1 1	0 0 1 0 0 0 0
1 0 0	0 0 0 1 0 0 0
1 0 1	0 0 0 0 1 0 0
1 1 0	0 0 0 0 0 1 0
1 1 1	0 0 0 0 0 0 1

One may notice from the previous table that for $\mathbf{S}^T = [101]$ the error word is $\mathbf{e} = [0000100]$. The correction is done as follows:

$$\mathbf{v} = \mathbf{r} \oplus \mathbf{e} = [1 1 0 1 1 0 1] \oplus [0 0 0 0 1 0 0] = [1 1 0 1 0 0 1]$$

e). Increasing the code distance from 3 to 4 can be done in two ways: extending or shortening the code.

Table 5.4 Standard array for the Hamming (7,4) code.

0000000	1101001	0101010	1000011	1001100	0100101	1100110	0101111	1110000	0011001	1011010	0110011	0111100	1010101	0010110	1111111
1000000	0101001	1101010	0000011	0001100	1100101	0100110	1101111	0110000	1011001	0011010	1110011	1111100	0010101	1010110	0111111
0100000	1001001	0001010	1100011	1101100	0000101	1000110	0001111	1010000	0111001	1111010	0010011	0011100	1110101	0110110	1011111
0010000	1111001	0111010	1010011	1011100	0110101	1110110	0111111	1100000	0001001	1001010	0100011	0101100	1000101	0000110	1101111
0001000	1100001	0100010	1001011	1000100	0101101	1101110	0100111	1111000	0010001	1010010	0111011	0110100	1011101	0011110	1110111
0000100	1101101	0101110	1000111	1001000	0100001	1100010	0101011	1110100	0011101	1011110	0110111	0111000	1010001	0010010	1111011
0000010	1101011	0101000	1000001	1001110	0100111	1100100	0101101	1110010	0011011	1011000	0110001	0111110	1010111	0010100	1111101
0000001	1101000	0101011	1000010	1001101	0100100	1100111	0101110	1110001	0011000	1011011	0110010	0111101	1010100	0010111	1111110

The $H(7,4)$ code extension is obtained using matrix \mathbf{H}^* given by (5.81) and a structure codeword (5.80).

$$\mathbf{H}^* = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}, \quad \mathbf{v}^* = [c_0 c_1 c_2 a_3 c_4 a_5 a_6 a_7]$$

The encoding relations for c_1, c_2, c_4 are identical to the basic $H(7,4)$ code, along with:

$$c_0 = c_1 \oplus c_2 \oplus a_3 \oplus c_4 \oplus a_5 \oplus a_6 \oplus a_7$$

Let us consider the following sequences are received:

$$\mathbf{r}_1 = 0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1$$

$$\mathbf{r}_2 = 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1$$

$$\mathbf{r}_3 = 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1$$

$$\mathbf{r}_4 = 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1$$

Determine, if possible, the correct sequences.

$$\mathbf{H}^* \mathbf{r}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \dots \\ 0 \end{bmatrix} \Rightarrow \mathbf{r}_1 \text{ is a codeword, meaning that no errors or the errors are}$$

not detectable.

$$\mathbf{H}^* \mathbf{r}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \dots \\ 1 \end{bmatrix} \Rightarrow \mathbf{S}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad S_2 = 1, \text{ it follows that there is a correctable error on the first position.}$$

It results that:

$$\mathbf{v} = \mathbf{r}_2 \oplus [0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0] = [0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1]$$

$$\mathbf{H}^* \mathbf{r}_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \dots \\ 1 \end{bmatrix} \Rightarrow \mathbf{S}_1 = 0, \quad S_2 = 1 \quad \text{therefore } c_0 \text{ is erroneous.}$$

It results:

$$\mathbf{v} = \mathbf{r}_3 \oplus [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0] = [0\ 1\ 1\ 0\ 1\ 0\ 0\ 1]$$

$$\mathbf{H}^* \mathbf{r}_4 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ \dots \\ 0 \end{bmatrix} \Rightarrow \mathbf{S}_1 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, S_2 = 0, \text{ it results that there are two uncorrect-}$$

able errors. However, the even errors have been detected.

Code shortening

It is well known that shortening a basic H(n,m) code is performed by reducing the information symbols number, so in our case, where m is fixed (4) we cannot start from the H(7,4) code but from the perfect H(15,11) code that follows immediately after:

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

$\times \quad \times \quad \times \quad \times \quad \times \quad \times \quad \times \quad \times$

Eliminating the columns that contain an even number of ones we obtain:

$$\mathbf{H}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

One may notice that this matrix has dimensions compatible to the data format (m = 4) and ensures d = 4 requirement. In this case the structure of the codeword is:

$$\mathbf{v}_0 = [c_1 \ c_2 \ c_3 \ a_4 \ c_5 \ a_6 \ a_7 \ a_8]$$

The encoding relations can be determined with:

$$\mathbf{H}_0 \mathbf{v}_0^T = \mathbf{0} \Leftrightarrow \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ a_4 \\ c_5 \\ a_6 \\ a_7 \\ a_8 \end{bmatrix} = \mathbf{0}$$

$$\Rightarrow \begin{cases} c_5 = a_6 \oplus a_7 \oplus a_8 \\ c_3 = a_4 \oplus a_7 \oplus a_8 \\ c_2 = a_4 \oplus a_6 \oplus a_8 \\ c_1 = a_4 \oplus a_6 \oplus a_7 \end{cases}$$

Let us consider that the following sequences are received:

$$\mathbf{r}_1 = [1 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0]$$

$$\mathbf{r}_2 = [0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0]$$

$$\mathbf{r}_3 = [0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0]$$

An estimation of the received sequences can be done checking the following equation:

$$\mathbf{H}_0 \mathbf{r}^T = \mathbf{S}$$

$$\mathbf{H}_0 \mathbf{r}_1^T = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \dots \\ 0 \end{bmatrix}, \text{ it follows that } \mathbf{r}_1 \text{ is either correct or the errors can not be}$$

detected ($\mathbf{S}_1 = 0, \mathbf{S}_2 = 0$).

$$\mathbf{H}_0 \mathbf{r}_2^T = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \dots \\ 1 \end{bmatrix}, \text{ it results that there is one error } (\mathbf{S}_1 = 0, \mathbf{S}_2 = 1).$$

The syndrome value shows that it is located on the position c_1 , so the correct word is:

$$\mathbf{v} = \mathbf{r}_2 \oplus [10000000] = [11110000]$$

$$\mathbf{H}_0 \mathbf{r}_3^T = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \dots \\ 1 \end{bmatrix}, \text{ the errors are double and can not be corrected}$$

($S_1 \neq 0, S_2 = 1$), however the errors have been detected.

5.7.10 Some Other Linear Block Codes

Simple parity check codes

These codes are among the most used error detection codes in data transmission systems; a simple parity code adds one supplementary bit (parity check bit) to the information bits of the code word. It follows that:

$$n = m + 1 \tag{5.84}$$

The parity check bit is determined depending on the used parity criterion:

Odd parity criterion:

$$\sum_{i=1}^n a_i = 1 \tag{5.85}$$

Even parity criterion:

$$\sum_{i=1}^n a_i = 0 \tag{5.86}$$

Remark

$\sum_{i=1}^n$ - is designating the modulo 2 summation

This code can detect all the odd errors. The code relative redundancy is:

$$R = \frac{k}{n} = \frac{1}{n}$$

Using the two parity criteria, determine the parity bit for the following sequence: 1011011.

Using the odd parity criterion: $1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus c = 1$; it results that $c = 0$.

Using the even parity criterion: $1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus c = 1$; it results that $c = 1$.

The codewords will be $[10110110]$ and $[10110111]$.

Cross parity check code

It is obtained iterating two simple parity check error detecting codes.

The information received from the transmitter is arranged in blocks of information sequences: a parity check bit is associated to each row and to each column (\mathbf{R}_i), resulting a transversal parity check word (Table 5.6).

Table 5.6 Encoding table for the cross parity check code.

	\mathbf{R}_1	\mathbf{R}_2	...	\mathbf{R}_m	C_{i0}
\mathbf{v}_1	i_{11}	i_{12}	...	i_{1m}	C_{10}
\mathbf{v}_2	i_{21}	i_{22}	...	i_{2m}	C_{20}

\mathbf{v}_p	i_{p1}	i_{p2}	...	i_{pm}	C_{p0}
C_{jv}	C_{1v}	C_{2v}	...	C_{mv}	

The receiver checks the parity for each received word and then builds a transversal parity check word of its own, which is compared to the received word. If the two words are identical and if the horizontal parity is checked for each word, the block is validated.

This code can correct any singular error occurred inside the block, due to the fact that this error affects both the horizontal and vertical parity check. There are some cases in which the horizontal and vertical errors can be detected. A detailed analysis of these particular cases is performed in [24]. This code was used for information protection in recording the information on magnetic tape or paper.

Example 5.9

Let us consider the block:

	\mathbf{R}_1	\mathbf{R}_2	\mathbf{R}_3	\mathbf{R}_4	\mathbf{R}_5	\mathbf{R}_6	\mathbf{R}_7	C_{i0}
\mathbf{v}_1	1	0	1	1	0	0	1	1
\mathbf{v}_2	0	1	1	0	1	1	0	1
\mathbf{v}_3	1	1	0	0	1	0	0	$0 \Rightarrow 1$
\mathbf{v}_4	0	0	1	0	1	0	1	0
\mathbf{v}_5	0	1	1	0	0	1	1	1
C_{kv}	1	0	1	$0 \Rightarrow 1$	0	1	0	1

The odd parity criterion was used for encoding. We assume that an error occurs on position $(\mathbf{v}_3, \mathbf{R}_4)$. It will affect C_{30} and C_{4v} , therefore it will be corrected.

Constant weight code (m, n)

Unlike all the codes we have already studied, the constant weight code is a non separable code, i.e. with implicit redundancy. The length of the code words is n . The encoding law stipulates that all the words have the same weight $w = m$, allowing the all odd errors detection. This code is used in information alpha numeric representations, for example code (3, 7) in telex.

We will determine the redundancy and detection capacity taking into account relation (5.10):

$$R_r = \frac{\lg 2^n - \lg C_n^m}{\lg 2^n} = \frac{n - \lg C_n^m}{n} \quad (5.87)$$

$$C_d = \frac{N_{te} - N_{ue}}{N_{te}} = 1 - \frac{N_{ue}}{N_{te}}$$

where $N_{ue} = C_n^m - 1$ represents the number of undetectable erroneous combinations and $N_{te} = 2^n - 1$ is the total number of possible erroneous words. The total codewords number is $M = C_n^m$. In this case C_d is expressed as:

$$C_d = 1 - \frac{C_n^m - 1}{2^n - 1} \quad (5.88)$$

Example 5.10

Let us consider the code $C(2, 5)$: 11000, 00011, 00110, 01010, 01100, 10001, 10010, 10100. It follows that maximum ten messages can be encoded ($M = C_5^2 = 10$).

The redundancy and code detection capacity are computed using (5.87) and (5.88).

$$R_r = \frac{5 - \lg 10}{5} = \frac{5 - 3,32}{5} = 0,336 = 33,6\%$$

$$C_d = 1 - \frac{10 - 1}{2^5 - 1} = \frac{22}{31} = 0,7 = 70\% .$$

5.8 Cyclic Codes

Cyclic codes are an important sub-set of linear block codes. They are very used in practice due to the simplicity in implementation using linear feedback shift registers (LFSR); the algebraic structure of these codes made possible a lot of practical decoding algorithms.

From the history of this codes we mention: 1957- E. Prange is the first to study the cyclic codes, 1959 – A. Hocquenghem and 1960 – R. Bose and P. Chauduri

are giving multiple errors correcting cyclic codes also known as BCH (Bose-Chaudhuri-Hocquenghem) codes, 1960 – J. Reed and G. Solomon give the cyclic non-binary codes known as RS (Reed-Solomon) codes. W. Peterson wrote a monography about cyclic codes in 1961. Other important contributions in developing these codes belong to: G. Forney, Y. Massey, R. Blahut, E. Berlekamp, T. Kasammi, L. Chen, S. Lin.

5.8.1 Definition and Representation

A linear block code $C(n, m)$ is cyclic if any cyclic permutation of a codeword is also a codeword.

If:

$$\mathbf{v} = (a_0 a_1 \dots a_{n-1}) \in C$$

then any cyclic permutation of \mathbf{v} is still a codeword:

$$\mathbf{v}^{(1)} = (a_{n-1} a_0 \dots a_{n-2}) \in C$$

...

$$\mathbf{v}^{(i)} = (a_{n-i} a_{n-i+1} \dots a_{n-1} a_0 \dots a_{n-i-1}) \in C$$

For cyclic codes the words are represented by polynomials; for an n symbols sequence the corresponding polynomial is of degree $n - 1$ or smaller:

$$v(x) = a_0 + a_1 x + a_2 x^2 \dots + a_{n-1} x^{n-1} \quad (5.89)$$

The encoding relation, giving the codewords set C , is to form codewords, polynomially represented as multiples of k -th degree polynomial, known as the *generator polynomial*.

From the cyclic codes definition we get that for any information polynomial $i(x)$ of degree $m - 1$, the codewords are chosen multiples of a $k = n - m$ degree polynomial known as the generator polynomial of the code.

$$i(x) = i_0 + i_1 x + \dots + i_{m-1} x^{m-1} \quad (5.90)$$

$$g(x) = g_0 + g_1 x + \dots + g_k x^k, \quad g_k = g_0 = 1 \quad (5.91)$$

$$v(x) = i(x)g(x) \quad (5.92)$$

All distinct combinations set that can be formed with n symbols (codewords) forms an algebra. This set comprises the set of the residue classes of an n degree polynomial $c(x)$ with coefficients in $GF(2)$.

The polynomial $c(x)$ of degree n can be chosen randomly, but for obtaining a similar expression with the scalar product between two vectors [43], [53] it is chosen as follows:

$$c(x) = x^n + 1 = 0 \quad (5.93)$$

For binary codes the set residue classes of $c(x) = x^n + 1 = 0$ has 2^n elements. From this algebra we can choose 2^m elements subset (the codewords set), that are multiples of the generator polynomial (the *ideal generated by $g(x)$*).

Due to the fact that the zero element belongs to the ideal, it results that exists a polynomial $h(x)$ such that

$$g(x) \cdot h(x) = c(x) = x^n + 1 = 0 \quad (5.94)$$

It follows that $g(x)$ is chosen from the divisors of $c(x) = x^n + 1 = 0$, hence:

$$h(x) = \frac{x^n + 1}{g(x)} \quad (5.95)$$

5.8.2 Algebraic Encoding of Cyclic Codes

The codeword formed with relation (5.92) leads to a non-systematic cyclic code (the information is modified in the encoded structure)

Relation (5.92) can be rewritten as follows:

$$\begin{aligned} v(x) &= i(x) \times g(x) = (i_0 + i_1x + \dots + i_{m-1}x^{m-1}) \times g(x) = \\ &= i_0g(x) + i_1xg(x) + \dots + i_{m-1}x^{m-1}g(x) \end{aligned} \quad (5.96)$$

Equation (5.96) shows that $v(x)$ is formed by the linear combinations set of the generator \mathbf{G} matrix line vectors, where:

$$\mathbf{G}_{[m \times n]} = \begin{bmatrix} \mathbf{g}(x) \\ x\mathbf{g}(x) \\ \vdots \\ x^{m-1}\mathbf{g}(x) \end{bmatrix} = \begin{bmatrix} g_0 & g_1 & \dots & g_k & 0 & \dots & 0 \\ 0 & g_0 & \dots & g_{k-1} & g_k & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & g_0 & g_1 & \dots & g_k \end{bmatrix} \quad (5.97)$$

In order to obtain a systematic structure, *i.e.* the information to be unmodified on the most significant positions, we follow the next steps:

1. $x^k i(x) = i_0x^k + i_1x^{k+1} + \dots + i_{m-1}x^{n-1}$
2. $\frac{x^k i(x)}{g(x)} = q(x) + \frac{r(x)}{g(x)}$
3. $\begin{aligned} v(x) &= x^k i(x) + r(x) = q(x)g(x) = \\ &= a_0 + a_1x + \dots + a_{k-1}x^{k-1} + a_kx^k + \dots + a_{n-1}x^{n-1} \end{aligned} \quad (5.98)$

Therefore we obtain a cyclic codeword, multiple of $g(x)$, with the information symbols placed on the most significant m positions: $x^{ki}(x)$ and k control symbols given by $r(x)$, the remainder after the division of $x^{ki}(x)$ to $g(x)$, a polynomial of maximum degree $k-1$.

The encoding relationship (5.98) used to obtain a systematic cyclic code can be rewritten in such a way to obtain an equation similar to $\mathbf{H}\mathbf{v}^T = 0$. Equation (5.98) is multiplied with $h(x)$ and due to the fact that $h(x)$ and $g(x)$ are orthogonal (relationship (5.94)) we get:

$$v(x)h(x) = q(x)g(x)h(x) = 0 \tag{5.99}$$

The product $v(x)h(x) = 0$ can be written as the scalar product between two vectors:

$$\left\{ \begin{array}{l} (a_0 a_1 \dots a_{n-1})(00 \dots 0 h_m h_{m-1} \dots h_1 h_0) = 0 \\ (a_0 a_1 \dots a_{n-1})(00 \dots h_m h_{m-1} h_{m-2} \dots h_1 h_0 0) = 0 \\ \vdots \\ (a_0 a_1 \dots a_{n-1})(h_m h_{m-1} \dots h_0 \dots 00) = 0 \end{array} \right. \tag{5.100}$$

Remark

From the n equations given by the cyclic permutations we have written only the $k = n - m$ linear independent ones.

The system (5.100) can be written also:

$$\begin{bmatrix} 0 & 0 & \dots & 0 & h_m & h_{m-1} & \dots & h_1 & h_0 \\ 0 & 0 & \dots & h_m & h_{m-1} & h_{m-2} & \dots & h_0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ h_m & h_{m-1} & \dots & h_0 & 0 & 0 & \dots & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix} = 0 \tag{5.101}$$

One may easily notice that this system is of $\mathbf{H}\mathbf{v}^T = 0$ type, where \mathbf{H} is identified as:

$$\mathbf{H}_{[k \times n]} = \begin{bmatrix} 0 & 0 & \dots & 0 & h_m & h_{m-1} & \dots & h_1 & h_0 \\ 0 & 0 & \dots & h_m & h_{m-1} & h_{m-2} & \dots & h_0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ h_m & h_{m-1} & \dots & h_0 & 0 & 0 & \dots & 0 & 0 \end{bmatrix} \tag{5.102}$$

Example 5.11

Let us consider code $C(7,4)$ with the generator polynomial $g(x) = x^3 + x + 1$.

According to (5.96), the generator matrix of this code is:

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

the polynomial $h(x)$ is determined from:

$$h(x) = \frac{x^7 + 1}{g(x)} = x^4 + x^2 + x + 1$$

where, according to (5.102), it results the control matrix structure.

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

Using relation (5.101) we determine the encoding relations:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} = 0 \Rightarrow \begin{cases} a_2 = a_4 + a_5 + a_6 \\ a_1 = a_3 + a_4 + a_5 \\ a_0 = a_2 + a_3 + a_4 \end{cases}$$

The entire discussion regarding cyclic codes encoding was done supposing the existence of a generator polynomial $g(x)$ of degree k . A question raises:

How should be $g(x)$ in order to correct a maximum number of t independent errors?

The answer can be given taking into consideration the finite fields theory (Galois and extended Galois fields [Appendix A]), which represent the mathematical support for cyclic codes.

From the encoding law, we know that $v(x)$ must be a multiple of $g(x)$, which, at its turn, according to (5.94), must divide $x^n + 1 = 0$.

In order to accomplish these requirements we proceed as follows:

- We choose r roots of $x^n + 1 = 0$ noted as:

$$\beta_i = \alpha^i \in \text{GF}(2^k), i = \overline{0, r-1} \text{ or } i = \overline{0, r} \quad (5.103)$$

These β_i roots are primitive elements (α) of the k -th order extension of the binary Galois field $GF(2^k)$. The order k of the extension is found from (Appendix A):

$$n = 2^k - 1 \Rightarrow k \text{ and } GF(2^k) \tag{5.104}$$

$GF(2^k)$ extension is generated by a primitive polynomial $p(x)$ of degree k determined with (5.104) (Table 5.7). Tables containing primitive polynomials for different degree until 100 can be found in Appendix A.9.

Table 5.7 Table of primitive polynomials up to degree $k = 5$.

k	a_k	$a_{k-1} \dots a_0$
1	1	1
2	1	1 1
3	1	0 1 1
		1 1 0 1
4	1	0 0 1 1
		1 1 0 0 1
5	1	0 0 1 0 1
		1 0 1 0 0 1
		1 0 1 1 1 1
		1 1 0 1 1 1
		1 1 1 1 0 1

Remark: k from equation (5.104) represents the extension order of $GF(2^k)$ field and not the number of the control symbols: $k = n - m$, which will be denoted as k' from this point on. As a consequence of this remark, there will be no misunderstandings regarding the use of k in these two cases.

- We determine $g(x)$ as the smallest common multiple of the minimal polynomials corresponding to β_i roots:

$$g(x) = \text{s.c.m}\{m_1(x), \dots, m_r(x)\} \tag{5.105}$$

If all the r polynomials are primes, then:

$$g(x) = \prod_{i=1}^r m_i(x) \tag{5.105.a}$$

A *minimal polynomial* corresponding to β_i is defined as the irreducible polynomial $m(x)$ of minimal degree for which $m_i(\beta_i) = 0$; it is obtained as:

$$m_i(x) = (x + \beta_i) \cdot \left(x + \beta_i^2 \right) \left(x + \beta_i^{2^{k-1}} \right) \tag{5.106}$$

- In order to obtain a t errors correcting code, we choose $r = 2t$ roots $\beta_i \in GF(2^k)$ and the code distance of the obtained code has the minimal distance $d \geq 2t + 1$ [Appendix A] [29] [52].
- Based on the theorem (Appendix A) that states: for binary cyclic codes, if α is a root $\alpha^2, \alpha^{2^2}, \dots, \alpha^{2^{k-1}}$ are also roots, to obtain the generator polynomial it is enough to select only the odd roots:

$$\beta_1 = \alpha, \beta_3 = \alpha^3, \dots, \beta_{2t-1} = \alpha^{2t-1} \tag{5.107}$$

In what follows we will enounce some theorems (without demonstration) useful in checking the calculus made for determining the polynomial $g(x)$ [2], [29]:

- The degree of any minimal polynomial $m_i(x)$ is smaller than or equal to k (the extension order)

$$\text{degree of } m_i(x) \leq k \tag{5.108}$$

- The degree of the generator polynomial $g(x)$ is smaller than or equal to $k \cdot t$ (we remind that the generator polynomial degree is equal to the control symbols.

$$k' = \text{degree of } g(x) = \text{control symbols number} = n - m \leq k \cdot t \tag{5.109}$$

in which k signifies the $GF(2^k)$ extension order

- The number of non-zero terms of $g(x)$ equal to the code distance:

$$\text{Non-zero terms } [g(x)] = d \geq 2t + 1 \tag{5.110}$$

The codes obtained as shown above (binary cyclic t errors correcting codes) are known as *BCH codes (Bose–Chauduri–Hocquenghem)*. Tables with generator polynomials for BCH codes for different n and t may be determined [28], [Appendix A]:

Table 5.8 BCH codes generator polynomials up to $n = 31$

n	m	t	g_k, g_{k-1}, \dots, g_0
7	4	1	13
	1	2	177
15	11	1	23
	7	2	721
	5	3	2467
31	1	7	77777
	26	1	45
	21	2	3551
	16	3	107657
	11	5	5423325
	6	7	313365047

Remark

The generator polynomial coefficients are given in octal for compression purposes.

BCH encoding can be done in several ways:

- using relation (5.92) for a non-systematic code
- using relation (5.101) in which the control matrix structure is determined imposing $v(x)$ to have as roots:

$$\beta_1 = \alpha, \beta_3 = \alpha^3, \dots, \beta_{2t-1} = \alpha^{2t-1}, \quad (5.111)$$

in this case we have:

$$v(\beta_i) = a_0 \beta_i^0 + a_1 \beta_i^1 + \dots + a_{n-1} \beta_i^{n-1} = 0 \quad (5.112)$$

$$i = 1, 3, 5, \dots, 2t - 1$$

We obtain:

$$\mathbf{H} = \begin{bmatrix} \alpha^0 \alpha^1 & \alpha^2 & \dots & \alpha^{n-1} \\ \alpha^0 \alpha^3 & \alpha^6 & \dots & \alpha^{3(n-1)} \\ \vdots & & & \\ \alpha^0 \alpha^{2t-1} & \alpha^{(2t-1)^2} & \dots & \alpha^{(2t-1)(n-1)} \end{bmatrix} \quad (5.113)$$

Remark

Each element α is expressed in k digits, therefore the control matrix dimension is $[tk \times n]$

- using relation (5.98), for a systematic structure $g(x)$ being determined as previously described, or chosen from available tables.

Example 5.12

Dimension the BCH error correcting codes of length $n = 15$ for $t = 1$, $t = 2$ and $t = 3$. Determine the generator polynomials and the encoding relationships. Determine the codewords for a random information sequence.

Solution

Galois field is dimensioned using (5.104)

$$n = 2^k - 1$$

$$15 = 2^k - 1 \Rightarrow k = 4 \Rightarrow \text{GF}(2^4)$$

The elements belonging to $\text{GF}(2^4)$ are the residues of polynomial $p(x)$, a primitive polynomial of degree $k = 4$. From Table 5.7 we choose:

$$p(x) = x^4 + x + 1$$

as the generator polynomial of $\text{GF}(2^k)$.

The primitive elements of the $GF(2^4)$ field can be represented using polynomials or matrices. We bear in mind that for any Galois field we have (Appendix A.5):

$$\alpha^n = 1$$

The elements of the $GF(2^4)$ field generated by $p(x) = x^4 + x + 1$ are given in Table 5.9.

For $t=1$ we choose $\beta_1 = \alpha \in GF(2^4)$. The minimal polynomial of this root is, according to (5.106):

$$m_1(x) = (x + \alpha)(x + \alpha^2)(x + \alpha^4)(x + \alpha^8) = x^4 + x + 1$$

and $g(x)$ according to (5.105) is:

$$g(x) = m_1(x) = x^4 + x + 1$$

One may notice that we obtained a polynomial of degree 4, so the number of control symbols will be $k' = 4 = k$, the order of the extension, because for $t = 1$ $g(x)$ is equal to $p(x)$. Code dimensions for $n = 15$ and $t = 1$ are: BCH(15,11,4) where $n=15$ represents the length of the codeword, $m = 11$ represents the information symbols, and $k'=4=n-m$ the control symbols.

Table 5.9 $GF(2^4)$ generated by $p(x) = x^4 + x + 1$

0 and α^i	Polynomial representation						Matrix representation
0	0						0000
1	1						1000
α		α					0100
α^2			α^2				0010
α^3				α^3			0001
α^4	1	$+$	α				1100
α^5			α	$+$	α^2		0110
α^6					α^2	$+$	α^3
α^7	1	$+$	α			$+$	α^3
α^8	1	$+$			α^2		1010
α^9			α	$+$			α^3
α^{10}	1	$+$	α	$+$	α^2		1110
α^{11}			α	$+$	α^2	$+$	α^3
α^{12}	1	$+$	α	$+$	α^2	$+$	α^3
α^{13}	1	$+$			α^2	$+$	α^3
α^{14}	1	$+$					α^3

The control matrix is expressed with (5.113)

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

The structure of a codeword is:

$$\mathbf{v} = \left[\underbrace{a_0 a_1 a_2 a_3}_{k'=4 \text{ control symbols}} \quad \underbrace{a_4 a_5 a_6 a_7 a_8 a_9 a_{10} a_{11} a_{12} a_{13} a_{14}}_{m=1 \text{ information symbols}} \right]$$

The encoding relation obtained with (5.101) are:

$$a_0 = a_4 \oplus a_7 \oplus a_8 \oplus a_{10} \oplus a_{12} \oplus a_{13} \oplus a_{14}$$

$$a_1 = a_4 \oplus a_5 \oplus a_7 \oplus a_9 \oplus a_{10} \oplus a_{11} \oplus a_{12}$$

$$a_2 = a_5 \oplus a_6 \oplus a_8 \oplus a_{10} \oplus a_{11} \oplus a_{12} \oplus a_{13}$$

$$a_3 = a_6 \oplus a_7 \oplus a_9 \oplus a_{11} \oplus a_{12} \oplus a_{13} \oplus a_{14}$$

In order to determine the codeword we choose the following information sequence:

$$\mathbf{i} = \left[\begin{array}{cccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \text{LSB} & & & & & & & & & & & \end{array} \right]$$

The codeword can be determined either using the encoding relations already found or using (5.98):

$$i(x) = 1$$

$$x^{k_i}(x) = x^4$$

$$x^{k_i}(x)/g(x) = x^4 / (x^4 + x + 1)$$

It results: $r(x) = x + 1$, so $v(x) = x^{k_i}(x) + r(x) = x^4 + x + 1$, or in polynomial representation:

$$\mathbf{v} = \left[\begin{array}{cccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ \text{MSB} & & & & & & & & & & & & & \end{array} \right]$$

For $t = 2$ we choose the following roots:

$$\beta_1 = \alpha \quad \text{and} \quad \beta_3 = \alpha^3$$

$$\mathbf{m}_1(x) = (x + \alpha)(x + \alpha^2)(x + \alpha^4)(x + \alpha^8) = x^4 + x + 1$$

$$\mathbf{m}_3(x) = (x + \alpha^3)(x + \alpha^6)(x + \alpha^{12})(x + \alpha^{24}) = x^4 + x^3 + x^2 + x + 1$$

It can be noticed that both minimal polynomials are of degree 4, meaning that (5.109) is fulfilled. The two polynomials being relatively primes, $g(x)$ is determined with (5.105.a) as:

$$g(x) = m_1(x)m_3(x) = x^8 + x^7 + x^6 + x^4 + 1$$

and in binary:

$$g = \left(\underbrace{111}_7 \underbrace{010}_2 \underbrace{001}_1 \right)$$

The corresponding octal notation is: $g = 721$.

Condition (5.110) is fulfilled for $g(x)$ too: $8 \leq 4 \times 2$. The non-zero terms of $g(x)$ is 5, so according to (5.25) the code corrects two errors (the calculus is correct). The code dimensions are $H(15,7,2)$ and the codeword structure is:

$$\mathbf{v} = \left[\underbrace{a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7}_{k'=8 \text{ control bits}} \underbrace{a_8 a_9 a_{10} a_{11} a_{12} a_{13} a_{14}}_{m=7 \text{ information bits}} \right]$$

The control matrix is:

$$H(15,7,2) = \begin{bmatrix} \alpha^0 & \alpha^1 & \alpha^2 & \dots & \alpha^{14} \\ \alpha^0 & \alpha^3 & \alpha^6 & \dots & \alpha^{52} \end{bmatrix}$$

We ask the reader, as an exercise, to determine the encoding relations, the dimensions of the correcting code for $t=3$ as well $g(x)$.

Choosing the following information sequence:

$$\mathbf{i} = \left[\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \text{MSB} \end{array} \right]$$

we determine the systematic codeword similar for $t=1$.

$$i(x) = 1$$

$$x^k i(x) = x^8$$

$$x^k i(x)/g(x) = x^8 / (x^8 + x^7 + x^6 + x^4 + 1)$$

$$v(x) = x^8 + x^7 + x^6 + x^4 + 1$$

$$\text{or } \left[\begin{array}{cccccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ \text{MSB} \end{array} \right]$$

5.8.3 Syndrome Calculation and Error Detection

Assuming we deal with additive errors, at receiver we get:

$$r(x) = v(x) + e(x) \quad (5.114)$$

The encoding rule is checked, i.e. $r(x)$ divides by $g(x)$:

$$\text{rem} \frac{r(x)}{g(x)} = s(x) = \text{rem} \frac{v(x)}{g(x)} + \text{rem} \frac{e(x)}{g(x)} = \text{rem} \frac{e(x)}{g(x)} \quad (5.115)$$

where $s(x)$ is the error syndrome.

From relation (5.115) it follows that:

- the syndrome does not depend on the transmitted codeword $v(x)$, but only on the error word $e(x)$;
- if $s(x) = 0$, it means that there are not errors or the errors are not detectable;
- if $s(x) \neq 0$, the error is detected;
- the syndrome $s(x)$ is a polynomial of maximum degree $k' - 1$, meaning that for binary codes there are $2^{k'}$ distinct combinations; it means that from $2^n - 1$ possible errors combinations, maximum $2^{k'} - 1$ will be corrected. (zero combination is used for showing the lack of errors).

Cyclic codes are the most used codes for detection of independent errors as well as burst errors.

In what follows we will express the cyclic codes detection capacity starting from the definition (5.11): $C_d = \frac{N_{ed}}{N_{te}} = \frac{N_{te} - N_{en}}{N_{te}} = 1 - \frac{N_{en}}{N_{te}}$, where N_{ue} is determined by the undetectable errors combinations, so, for which the error word is a multiple (M) of $g(x)$:

$$e(x) = M \cdot g(x)$$

Let us consider a burst of errors of length p , located between positions i and j : $p = j - i$

$$e(x) = e_j x^j + \dots + e_i x^i = x^i (e_j x^{j-i} + \dots + e_i) \quad (5.116)$$

Further on, we analyze the three situations that may arise: p smaller, equal and higher than the generator polynomial degree, k' .

- $p < k'$ means that $e(x)$ cannot be a multiple of $g(x)$, so $N_{ue} = 0$; it results $C_d = 1$; all burst errors, having lengths smaller than k' will be detected.
- if $p = k'$, there is only one error combination that may correspond to the generator polynomial:

$$e(x) = g(x), \text{ so } N_{ue} = 1$$

In this case

$$\begin{aligned} N_{te} &= 2^{p-1}, \text{ because } e_i = e_j = 1 \\ C_d &= 1 - \frac{1}{2^{p-1}} = 1 - 2^{-p+1} = 1 - 2^{p-k'} \end{aligned} \quad (5.117)$$

- if $p > k'$, the errors multiples of $g(x)$ will not be detected:

$$e(x) = M \cdot g(x) = m(x) \cdot g(x)$$

where $m(x)$ is a polynomial of maximum degree $p - k'$; we calculate:

$$\begin{aligned} N_{ue} &= 2^{p-k'-1} \\ N_{te} &= 2^{p-1} \\ C_d &= 1 - \frac{2^{p-k'-1}}{2^{p-1}} = 1 - 2^{-k'} \end{aligned} \quad (5.118)$$

Remark

Cyclic codes detection capacity depends only on the generator polynomial degree, no other condition being imposed on $g(x)$. The notations used are: $n = m + k'$, where n is the length of the codeword, m the length of the information block and k' is the number of redundant symbols, equal to the degree of the generator polynomial.

Cyclic codes for error detection

Cyclic codes are the most used codes in ARQ systems. A block of data forming a message $M(x)$ is cyclically encoded using a generator polynomial of degree k' determined by the number of detectable errors. At receiver the block $r(x)$ is divided to $g(x)$. If the remainder is non zero, the error detection is performed and the retransmission is requested. This technique extremely often used in practice is known as *CRC (Cyclic Redundancy Check)*.

Some of its applications are given below:

- In data transmissions, the CCITT V41 notification recommends CRC with $g(x) = x^{16} + x^{12} + x^5 + 1$. Data block lengths are $n = 260, 500, 900, 3860$. The control symbols (CRC sequence) are two bytes long (15 degree polynomials). In the IBM/360 network, the polynomial used in CRC-16 is $g(x) = x^{16} + x^{15} + x^2 + 1$.
- Error protection in digital telephony - European Standard for data synchronization is done with CRC-4.
- 22469/1-80 standard, equivalent to ISO 4335-79 for data transmission with high level connection control procedures (HDLC-High Data Link Control), establishes the message and check sequence structure (Fig. 5.10) which is explained further on.

The m information bits (data + command + address) are represented by the polynomial $i(x)$ and are included between the delimitation byte and FCS. The control sequence CRC-16 in the HDLC frame is called "frame checking sequence" and denoted with FCS (Frame Checking Sequence). The sequences formed by two control bytes at the frame level FLC (Frame Level Control) are used for protocol implementation at link level.

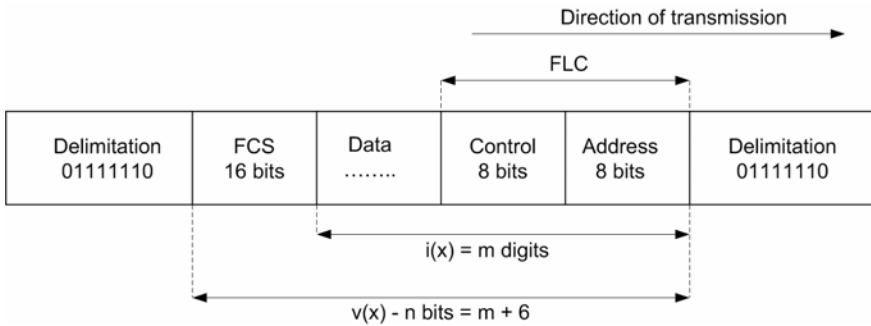


Fig. 5.10 HDLC frame format

The delimitation bytes are giving the length of the frame. If a delimitation sequence is not placed correctly between two frames, the receiver will assign them as a single frame. If the reception of the first frame is error free, the state of the LFSR (used for cyclic encoding and decoding) is 0, so the receiver will not detect the lack of the delimitation sequence for separating the frames. This problem occurs because the LFSR state is the same (zero) before a frame error free reception and after its check. To avoid this problem we need to modify the FCS so that the two configurations of the LFSR are different for the two previously mentioned cases.

The generator polynomial $g(x)$ is used for dividing the polynomial $M(x)$:

$$g(x) = x^{16} + x^{12} + x^5 + 1 \tag{5.119}$$

$$M(x) = x^{16}i(x) + x^mL(x) \tag{5.120}$$

where:

$$L(x) = \sum_{j=0}^{15} x^j \tag{5.121}$$

is used for the inversion of the first 16 most significant bits from $x^{16}i(x)$, determining the initialisation of the LFSR with “1”, in all binary positions. At the transmitter the original message $i(x)$ is transmitted and concatenated with the complemented remainder of the division of $x^k i(x)$ to $g(x)$ (FCS):

$$v(x) = x^{16}i(x) + \text{FCS} \tag{5.122}$$

where FCS is the complement with respect to “1” of the remainder $r(x)$ obtained from:

$$\frac{x^{16}i(x) + x^mL(x)}{g(x)} = q(x) + \frac{r(x)}{g(x)} \tag{5.123}$$

so

$$\text{FCS} = \overline{r(x)} = r(x) + L(x) \tag{5.124}$$

Adding the polynomial $x^m L(x)$ to $x^{16}i(x)$ is equivalent to setting the initial remainder as 1. By complementing with respect to 1 the $r(x)$ by the transmitter, after performing the division, the error free received message will generate a unique non zero remainder; this ensures the protection in the case when there is no delimitation sequence at the message end.

At receiver the division is done:

$$\begin{aligned} \frac{x^{16}v(x) + x^{16+m}L(x)}{g(x)} &= \frac{x^{16} \left[x^{16}i(x) + x^m L(x) + r(x) \right]}{g(x)} + \frac{x^{16}L(x)}{g(x)} = \\ &= x^{16}q(x) + x^{16} \frac{L(x)}{g(x)} \end{aligned} \tag{5.125}$$

If the transmission is not affected by errors, the receiver remainder is exactly $\frac{x^{16}L(x)}{g(x)}$, which is $x^{12} + x^{11} + x^{10} + x^8 + x^3 + x^2 + x + 1$.

Using this method, the transmitter and the receiver, invert the first 16 most significant bits by initialising the LFSR with “1” in all binary position. The remainder, after the reception of an error free block, is different from 0 and it has the configuration: 0 0 0 1 1 1 0 1 0 0 0 0 1 1 1 1, allowing the receiving

MSB

of two correct frames with their FCS concatenated because of the lack of delimitation sequence, without being considered as a unique frame.

5.8.4 Algebraic Decoding of Cyclic Codes

In what follows we will present the most used algorithm for cyclic decoding: Peterson algorithm with Chien search [47].

As shown in 5.8.2, a cyclic t errors correcting code has codewords (v_i) with $r = 2t$ roots $\beta_i \in GF(2^k)$, $\beta_i = \alpha^i$ as given in (5.112).

For BCH codes we take into consideration only the odd roots: $i = \overline{1, 2t-1}$ (see relation (5.111)).

At receiver, the encoding relation is checked (5.112), which, for additive errors is written as follows:

$$r(\beta_i) = v(\beta_i) + e(\beta_i) = e(\beta_i) = e(\alpha^i) = S_i, i = 1, 3, \dots, 2t-1 \tag{5.126}$$

For $2t$ roots, the syndrome S is:

$$S = (S_1 S_3 \dots S_{2t-1}) \tag{5.127}$$

For binary codes it is sufficient to determine the error positions from the syndrome.

Let us now see how S_i indicates the error position. For this we take randomly an error-word, with two errors, on positions 2 and 4:

$$e(x) = x^2 + x^4$$

The syndrome S , according to (5.126), is:

$$S_i = e(\alpha^i) = (\alpha^i)^2 + (\alpha^i)^4 = (\alpha^2)^i + (\alpha^4)^i = X_1^i + X_2^i$$

where by X_k we denoted the *error locator*, expression that indicates the error position:

$$X_k = \alpha^j, \quad k = \overline{1, t} \quad \text{and} \quad j = \overline{0, n-1} \quad (5.128)$$

Therefore, it can be seen that the syndrome S_i can be expressed as:

$$S_i = \sum_{k=1}^t X_k^i \quad (5.129)$$

which means that the error positions determination is nothing else but solving a system of non-linear equations with unknowns X_k . There are numerous methods to solve non-linear equation systems and all of them could be algebraic decoding algorithms for cyclic codes. In what follows, we will show one of the first and most efficient decoding algorithms: *Peterson algorithm with Chien search*.

Remark

Encoding and decoding for cyclic codes can be done in time as well as in frequency. Now we deal with the time approach, continuing to discuss the second method for Reed–Solomon codes (RS).

Peterson algorithm with Chien search has the following steps:

1. Error syndrome calculation

$$S = (S_1, S_3, \dots, S_{2t-1}); \quad S_i = r(\alpha^i) = \sum_{k=1}^t X_k^i, \quad i = 1, 3, \dots, 2t-1$$

2. Finding the *error polynomial* $\sigma(x)$ with roots the X_k locators.

$$\sigma(x) = \prod_{k=1}^t (x + X_k) = x^t + \sigma_1 x^{t-1} + \dots + \sigma_t \quad (5.130)$$

The coefficients σ_1 will be determined taking into account the S_i syndromes previously calculated.

The X_k locators are the roots of $\sigma(x)$ polynomial, so:

$$\sigma(X_k) = 0, \quad \forall k = \overline{1, t} \quad (5.131)$$

Replacing in (5.131) x with \mathbf{X}_k we obtain:

$$\mathbf{X}_k^t + \sigma_1 \mathbf{X}_k^{t-1} + \dots + \sigma_t = \mathbf{0} \quad (5.132)$$

Multiplying (5.132) with \mathbf{X}_k^i and summing for $k = \overline{1, t}$, we have:

$$\sum_{k=1}^t \mathbf{X}_k^{t+i} + \sigma_1 \sum_{k=1}^t \mathbf{X}_k^{t+i-1} + \dots + \sigma_t \sum_{k=1}^t \mathbf{X}_k^i = \mathbf{0} \quad (5.133)$$

Identifying \mathbf{S}_i (relationship (5.129)) we can write:

$$\mathbf{S}_{t+i} + \sigma_1 \mathbf{S}_{t+i-1} + \dots + \sigma_t \mathbf{S}_i = \mathbf{0}, \quad i = \overline{1, t} \quad (5.134)$$

It can be noticed that (5.134) is, in fact, a linear system of t equations with t unknowns. We can solve this system applying Cramer rule:

- Compute the determinant:

$$D = \begin{vmatrix} \mathbf{S}_t & \mathbf{S}_{t-1} & \dots & \mathbf{S}_1 \\ \mathbf{S}_{t+1} & \mathbf{S}_t & \dots & \mathbf{S}_2 \\ \vdots & & & \\ \mathbf{S}_{2t-1} & \mathbf{S}_{2t} & \dots & \mathbf{S}_t \end{vmatrix} \quad (5.135)$$

- If $D \neq 0$, then the system has a unique solution, given by:

$$\sigma_t = \left(\frac{D_1}{D}, \dots, \frac{D_j}{D}, \dots, \frac{D_t}{D} \right) \quad (5.136)$$

where D_j , $j = \overline{1, t}$, are characteristic determinants, obtained replacing in D the column j with the column formed by the free terms of the system (5.134).

Remarks

- If $D = 0$, the solution is not determined, it is supposed that the received word contains less than t errors.
- If $D = 0$, we search for an integer as large as possible, but $e \leq t$, such that $D_e \neq 0$; it is supposed that in this case, the transmission was affected by e errors.
- If such an integer e does not exist, we may say that the transmission was not affected by errors; this case is easily illustrated by $\mathbf{S}_i = \mathbf{0}$, $i = \overline{1, 2t-1}$.
- For the BCH codes we must take into account that, according to theorem (5.108), $\mathbf{S}_{2k} = \mathbf{S}_k^2$.

Table 5.10 Coefficients of the error polynomials for BCH codes.

t	σ_i
1	$\sigma_1 = S_1$
2	$\sigma_1 = S_1$ $\sigma_2 = (S_3 + S_1^3)/S_1$
3	$\sigma_1 = S_1$ $\sigma_2 = (S_1^2 S_3 + S_5)/(S_1^3 + S_3)$ $\sigma_3 = (S_1^3 + S_3) + S_1 \sigma_2$

Table 5.10 contains the coefficients $\sigma_i = f(S_i)$ for $t=1, 2$ and 3 ; the table can be easily filled in for $t > 3$ (practically up to $t = 5$, the limit case for a practical usage of the algorithm).

3. Chien Search

Chien search algorithm indicates the error position at the moment when one erroneous symbol reaches the last cell of a memory register in which the received word is loaded.

If the erroneous position is X_k , the error polynomial is given by (5.132). Dividing (5.132) by X_k^t we obtain:

$$\sum_{i=1}^t \sigma_i X_k^{-i} = 1 \quad (5.137)$$

index i showing the erroneous position. The error may occur on all n position, the maximum number of correctable errors being t .

In Chien search, the search of erroneous symbol begins with r_{n-1} , and in this case X_k is replaced with $\alpha^{(n-1)}$.

$$\alpha^{-i(n-1)} = \alpha^{-i \cdot n} \cdot \alpha^{i \cdot 1} = \left(\frac{1}{\alpha^n}\right)^i \alpha^{i \cdot 1} = \alpha^i$$

The symbol r_{n-j} will occur in the search equation (5.137) as follows:

$$\alpha^{-i(n-j)} = \alpha^{-i \cdot n} \cdot \alpha^{i \cdot j} = \alpha^{i \cdot j}$$

Finally, Chien search equation (5.137) will be:

$$\sum_{i=1}^t \sigma_i \alpha^{i \cdot j} = 1, \quad j = \overline{1, n} \quad (5.138)$$

where to index $j = 1$ corresponds r_{n-1} and to $j = n$ corresponds r_0 (the reception is done beginning with r_{n-1}).

The j value for which equation (5.138) is satisfied will provide \mathbf{X}_k

$$\mathbf{X}_k = \mathbf{a}^{n-j} \quad (5.139)$$

Example 5.13

Let us assume the reception of a BCH(15,7) word:

$$\mathbf{r} = \left[\begin{array}{ccccccccccccccc} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array} \right]_{\text{MSB}}$$

Using Peterson algorithm with Chien search find the decoded sequence.

Solution

$$r(x) = x^{14} + x^{12} + x^8 + x^7 + x^6 + x^4 + 1$$

We apply the presented algorithm:

- Calculating the error syndrome:

$$\mathbf{S}_1 = r(\mathbf{a}) = \mathbf{a}^{14} + \mathbf{a}^{12} + \mathbf{a}^8 + \mathbf{a}^7 + \mathbf{a}^6 + \mathbf{a}^4 + 1 = \mathbf{a}^5$$

$$\mathbf{S}_3 = r(\mathbf{a}^3) = \mathbf{a}^{42} + \mathbf{a}^{36} + \mathbf{a}^{24} + \mathbf{a}^{21} + \mathbf{a}^{18} + \mathbf{a}^{12} + 1 = \mathbf{a}^4$$

- Determining the coefficients σ_i :

From table 5.10, for $t=2$, we obtain the formulas for σ_1 and σ_2 :

$$\sigma_1 = \mathbf{S}_1 = \mathbf{a}^5$$

$$\sigma_2 = \frac{\mathbf{S}_3 + \mathbf{S}_1^3}{\mathbf{S}_1} = \frac{\mathbf{a}^4 + \mathbf{a}^{15}}{\mathbf{a}^5} = \frac{\mathbf{a}}{\mathbf{a}^5} = \mathbf{a}^{11}$$

- Chien search beginning with \mathbf{r}_{14} using relationship (5.138):

$$j=1 \rightarrow \sigma_1 \mathbf{a}^{1 \cdot 1} + \sigma_2 \mathbf{a}^{2 \cdot 1} = \mathbf{a}^5 \cdot \mathbf{a} + \mathbf{a}^{11} \cdot \mathbf{a}^2 = \mathbf{1},$$

This means that the symbol \mathbf{r}_{n-j} is erroneous, so $\mathbf{r}_{15-1} = \mathbf{r}_{14}$ is erroneous.

$$j=2 \rightarrow \sigma_1 \mathbf{a}^{1 \cdot 2} + \sigma_2 \mathbf{a}^{2 \cdot 2} = \mathbf{a}^9 \neq \mathbf{1}$$

$$j=3 \rightarrow \sigma_1 \mathbf{a}^{1 \cdot 3} + \sigma_2 \mathbf{a}^{2 \cdot 3} = \mathbf{a}^5 \cdot \mathbf{a}^3 + \mathbf{a}^{11} \cdot \mathbf{a}^6 = \mathbf{a}^8 + \mathbf{a}^{17} = \mathbf{1},$$

so $\mathbf{r}_{n-j} = \mathbf{r}_{15-3} = \mathbf{r}_{12}$ is erronated

$$j=4 \rightarrow \sigma_1 \mathbf{a}^{1 \cdot 4} + \sigma_2 \mathbf{a}^{2 \cdot 4} \neq \mathbf{1}$$

$$j=5 \rightarrow \sigma_1 \mathbf{a}^{1 \cdot 5} + \sigma_2 \mathbf{a}^{2 \cdot 5} \neq \mathbf{1}$$

⋮

$$j=14 \rightarrow \sigma_1 \mathbf{a}^{1 \cdot 5} + \sigma_2 \mathbf{a}^{2 \cdot 5} \neq \mathbf{1}$$

It follows that the error word is:

$$\mathbf{e} = [1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$$

The decoded word will be:

$$\begin{aligned} \mathbf{v} &= \mathbf{r} \oplus \mathbf{e} = [1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 1] + \\ &\quad + [1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0] = \\ &= \left[\begin{array}{c} 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 1 \\ \text{MSB} \end{array} \right] \end{aligned}$$

The information sequence at the decoder output is:

$$\mathbf{i} = [0\ 0\ 0\ 0\ 0\ 0\ 1].$$

5.8.5 Circuits for Cyclic Encoding and Decoding

As shown in 5.8.1 and 5.8.3, coding and decoding for systematic cyclic codes are done by division: $x^k i(x)$ and $r(x)$ respectively, to $g(x)$.

In what follows we will focus on two methods of dividing the polynomials, implying two methods of coding and decoding: using *linear feedback shift registers* (LFSR) with external or internal modulo two adders.

LFSR with external modulo two adders

A LFSR is a linear sequential circuit that can operate independently, without any input signal except the feedback signal. The register connections depend on the characteristic (generator) polynomial:

$$g(x) = g_k x^k + g_{k-1} x^{k-1} + \dots + g_1 x + g_0$$

in which $g_i \in \{0;1\}$ except for g_k which is always “1”. The block scheme of a LFSR with external modulo two adders is given in figure 5.11.

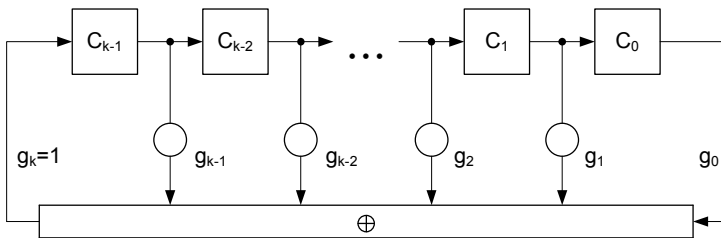


Fig. 5.11 LFSR with external modulo 2 adders

The register operation is described by the equations (5.140), where $st_i C_j$, $st_{i-1} C_j$ respectively, are the cell C_j states at the moment i , $i-1$ respectively.

$$\begin{cases} st_i C_0 = st_{i-1} C_1 \\ st_i C_1 = st_{i-1} C_2 \\ \vdots \\ st_i C_{k-1} = g_0 st_{i-1} C_0 + g_1 st_{i-1} C_1 + \dots + g_{k-1} st_{i-1} C_{k-1} \end{cases} \quad (5.140)$$

In a matrix description we have:

$$\mathbf{S}_i = \mathbf{T} \mathbf{S}_{i-1} \quad (5.140.a)$$

where

$$\mathbf{S}_i = \begin{bmatrix} st_i C_0 \\ \vdots \\ st_i C_{k-1} \end{bmatrix}, \quad \mathbf{S}_{i-1} = \begin{bmatrix} st_{i-1} C_0 \\ \vdots \\ st_{i-1} C_{k-1} \end{bmatrix}, \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ g_0 & g_1 & g_2 & \dots & g_{k-1} \end{bmatrix} \quad (5.141)$$

\mathbf{T} being the register characteristic matrix.

If the initial state of the LFSR is $\mathbf{S}_0 \neq 0$, its evolution in time will be the following: $\mathbf{S}_0, \mathbf{T} \mathbf{S}_0, \dots, \mathbf{T}^n \mathbf{S}_0 = \mathbf{S}_0$, so after a number of n steps it loads again the initial state; n is the register period. In order to have distinct states, \mathbf{T}^{-1} must exist, this being equivalent to have $g_0 = 1$. The total number of $2^k - 1$ non-zero states of the LFSR can be generated in one cycle or more.

The characteristic polynomial of the matrix \mathbf{T} is defined by:

$$\begin{aligned} \Phi(x) = \det[\mathbf{T} - x\mathbf{I}] &= \begin{vmatrix} -x & 1 & 0 & \dots & 0 & 0 \\ 0 & -x & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -x & 1 \\ g_0 & g_1 & g_2 & \dots & g_{k-1} & -x \end{vmatrix} = \\ &= g_0 + g_1 x + \dots + g_{k-1} x^{k-1} + x^k \end{aligned} \quad (5.142)$$

It can be seen that the characteristic polynomial of the matrix \mathbf{T} is the generator polynomial $g(x)$ such that the LFSR is uniquely determined.

The *characteristic matrix* \mathbf{T} is a root of the generator polynomial:

$$g(\mathbf{T}) = 0 \quad (5.143)$$

The *period of the characteristic matrix T*, in other words the cycle length, is the smallest integer n for which:

$$\mathbf{T}^n = \mathbf{T}^0 = \mathbf{I} \tag{5.144}$$

If **U** is the matrix:

$$\mathbf{U} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \tag{5.145}$$

The n non-zero states of the LFSR are:

$$\mathbf{T}^0\mathbf{U} = \mathbf{U}, \mathbf{T}\mathbf{U}, \mathbf{T}^2\mathbf{U}, \dots, \mathbf{T}^n\mathbf{U} = \mathbf{U} \tag{5.146}$$

Using the notation:

$$\mathbf{a}^i = \mathbf{T}^i\mathbf{U} \tag{5.147}$$

and if **T** is a root of *g(x)* and **aⁱ** is a root of *g(x)*, we get:

$$g(\mathbf{a}^i) = g(\mathbf{T}^i\mathbf{U}) = 0 \tag{5.148}$$

From the theory of residues modulo *g(x)* of degree k, we know that n is maximum if *g(x)* is primitive (**T** is called a primitive element of the field GF(2^k) generated by *g(x)* of degree k, primitive)

$$n = 2^k - 1$$

It follows that the *LFSR generates all the non-zero states in one cycle*, under the assumption that *g(x)* is primitive.

LFSR with internal modulo two adders

The block scheme of LFSR with internal modulo 2 adders is given in Fig. 5.12.

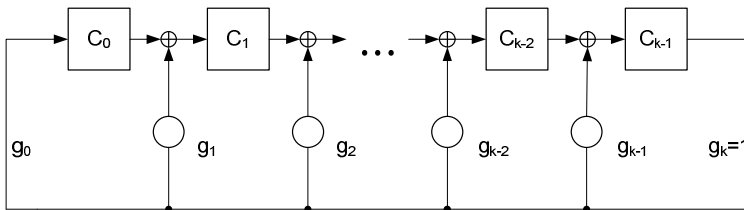


Fig. 5.12 LFSR with internal modulo 2 adders

In this case the characteristic matrix is:

$$\mathbf{T} = \begin{bmatrix} g_{k-1} & 1 & 0 & \dots & 0 \\ g_{k-2} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ g_1 & 0 & 0 & \dots & 1 \\ g_0 & 0 & 0 & \dots & 0 \end{bmatrix} \tag{5.149}$$

LFSRs from Fig. 5.11 and Fig. 5.12 are equivalents as they have the same characteristic polynomial.

Encoders for systematic cyclic codes implemented with LFSR

The process of encoding systematic cycle codes can be realized using LFSR with internal or external modulo two adders.

Encoder with LFSR

The block scheme of a cyclic encoder with LFSR (with external \oplus) is shown in Fig 5.13:

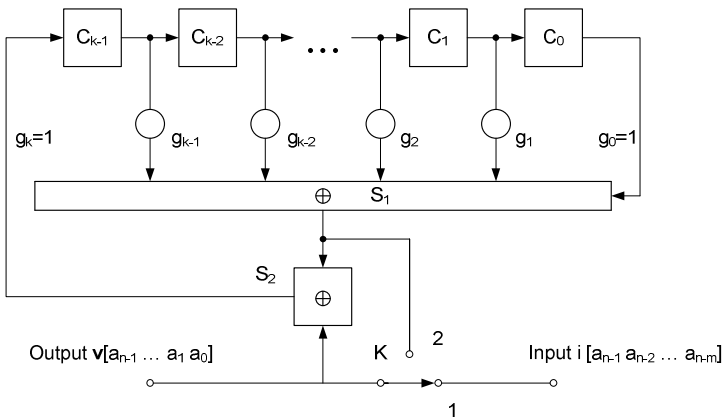


Fig. 5.13 Cyclic systematic encoder with LFSR and external modulo two adders

Comparing this scheme with Fig. 5.11 it can be easily observed that there is an input signal for the information symbols: $i [a_{n-1} \dots a_{n-m}]$, a modulo two adder S_2 and the switch K.

The input signal modifies the LFSR operating equation as follows:

$$S_i = \mathbf{TS}_{i-1} + a_i \mathbf{U} \tag{5.150}$$

where a_i is the input symbol at the moment i .

The switch K is on position 1 for m clocks, while the information symbols are delivered to the register; after m clocks it switches to position 2 for k clocks and the LFSR calculates the control symbols dividing $x^{k_i(x)}$ to $g(x)$. At the output we obtain the systematic cyclic code word: $v(x) = x^{k_i(x)} + r(x)$, where $r(x)$ is the remainder of $x^{k_i(x)}/g(x)$.

Due to the fact that, during the last k clocks the switch K is on position 2, the S_2 output will be 0 (its two input signals are identical). This means that at the end of n clocks the LFSR will be in the 0 state (all the cells will be 0).

Table 5.11 Cyclic encoder with LFSR and external modulo two adders

Ck	K	input i	S_i	Output v
1	1	a_{n-1}	$a_{n-1}U$	a_{n-1}
2		a_{n-2}	$a_{n-2}U + a_{n-1}TU$	a_{n-2}
...		...		
m		a_{n-m}	$a_{n-m}U + \dots + a_{n-1}T^{m-1}U$	a_{n-m}
m+1	2		$a_{n-m-1}U + \dots + a_{n-1}T^mU$	a_{n-m-1}
...				
m+k = n			$a_0U + a_1TU + \dots + a_{n-1}T^{n-1}U$	a_0

As it was already explained, at moment n the LFSR state will be zero:

$$S_n = 0 = [a_0U + a_1TU + \dots + a_{n-1}T^{n-1}U] \tag{5.151}$$

Relation (5.151) can also be written as a matrix product:

$$[U \quad TU \quad \dots \quad T^{n-1}U] \cdot \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix} = 0 \tag{5.151.a}$$

and more generally:

$$H \cdot v^T = 0$$

By identification we get for the control matrix the following structure:

$$H = [U \quad TU \quad \dots \quad T^{n-1}U] \tag{5.152}$$

where $U = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$ and T is determined by (5.141).

Encoder with LFSR and internal modulo two adders

The block scheme of the cyclic systematic encoder with LFSR and internal modulo two adders is given in figure 5.14.

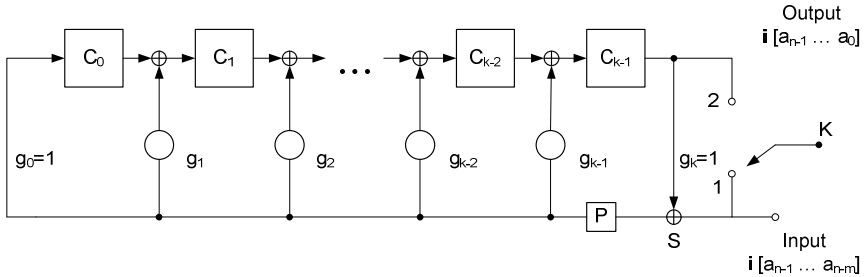


Fig. 5.14 Cyclic systematic encoder with LFSR and internal modulo two adders

Comparing this scheme with the one from Fig. 5.12, there are some changes: the modulo two adder S which allows us to supply the encoder with information symbols during the first m time periods, the gate P which is open during the first m periods and blocked during the last k periods, the switch K which is on position 1 during the first m clocks and on 2 during the last k clocks.

The encoder operation is given by relation (5.150):

$$S_i = TS_{i-1} + a_i U$$

At the end of the m clocks (as long we have information symbols) the register cells will be loaded with the remainder $[x^k i(x)/g(x)] = r(x)$; it follows that after the next k clocks the register will be in the zero state:

$$S_n = \mathbf{0}.$$

Similar to the case described in figure 5.13, the encoding relation $S_n = \mathbf{0}$ leads to $\mathbf{H} \cdot \mathbf{v}^T = \mathbf{0}$; at this point we identify \mathbf{H} with expression (5.152).

Example 5.14

We will illustrate a systematic cyclic encoder with LFSR and external modulo two adders, for $H(7,4)$ and $g(x) = x^3 + x + 1$. We will calculate the encoding relations

and solve them for $\mathbf{i} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ \text{MSB} \end{bmatrix}$.

For a given sequence \mathbf{i} , the systematic cyclic codeword, according to (5.98), is:

$$i(x) = 1$$

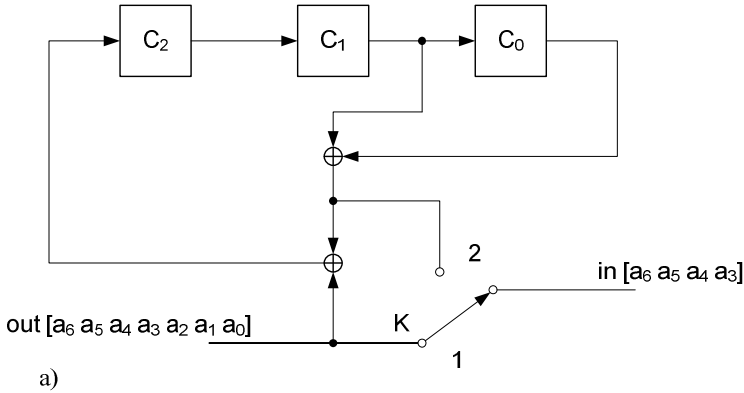
$$x^k i(x) = x^3$$

$$r(x) = \text{rem} \frac{x^3}{x^3 + x + 1} = x + 1, \text{ so } v(x) = x^3 + x + 1 \text{ or}$$

in a matrix form:

$$\mathbf{v} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ \text{MSB} \end{bmatrix}$$

The block scheme of a cyclic encoder with LFSR and external modulo two adders LFSR and its operation are given in figure 5.15.



		t_n	t_{n+1}			t_n
C_k	K	input i	C_2	C_1	C_0	output v
1	1	a_6	0	0	0	a_6
2		a_5	a_5	a_6	0	a_5
3		a_4	a_4+a_6	a_5	a_6	a_4
4		a_3	$a_3+a_5+a_6$	a_4+a_6	a_5	a_3
5	2		0	$a_3+a_5+a_6$	a_4+a_6	$a_2=a_4+a_5+a_6$
6			0	0	$a_3+a_5+a_6$	$a_1=a_3+a_4+a_5$
7			0	0	0	$a_0=a_3+a_5+a_6$

b)

Fig. 5.15 Cyclic encoder with LFSR and external \oplus : a) block scheme; b) operation table for $g(x)=x^3+x+1$ and $m=4$.

The scheme from Fig. 5.15.a) operates according to table from Fig. 5.15.b); we have already seen that equation $\mathbf{H} \cdot \mathbf{v}^T = \mathbf{0}$ describes the scheme behaviour where \mathbf{H} is given by (5.152) and \mathbf{T} and \mathbf{U} are by (5.141) and (5.145).

The register characteristic matrix \mathbf{T} is:

$$\mathbf{T} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ g_0 & g_1 & g_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \text{ and } \mathbf{U} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

It follows easily **H**:

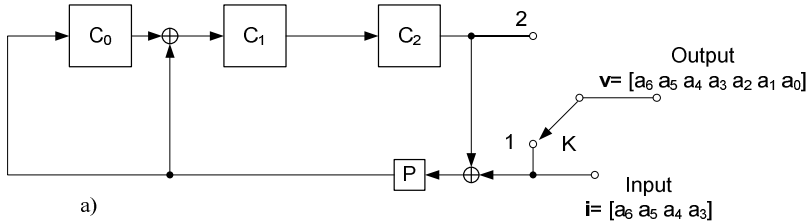
$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \text{ and then, } \mathbf{H}\mathbf{v}^T = \mathbf{0} \text{ leads to:}$$

$$\begin{cases} a_2 = a_4 \oplus a_5 \oplus a_6 \\ a_1 = a_3 \oplus a_4 \oplus a_5 \\ a_0 = a_2 \oplus a_3 \oplus a_4 = a_4 \oplus a_5 \oplus a_6 \oplus a_3 \oplus a_4 = a_3 \oplus a_5 \oplus a_6 \end{cases},$$

obviously identical with those previously determined.

For $\mathbf{i} = [0 \ 0 \ 0 \ 1]$ one can easily check (either using the operation table or taking into account the encoding system) that $\mathbf{v} = [0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1]$.

The block scheme of a cyclic encoder with LFSR and internal modulo two adders and its operation table are given in Fig. 5.16.



t_n			t_{n+1}			t_n
Ck	input i	P	C ₀	C ₁	C ₂	Output v
1	a ₆		a ₆	a ₆	0	a ₆
2	a ₅	ON	a ₅	a ₅ + a ₆	a ₆	a ₅
3	a ₄		a ₄ + a ₆	a ₄ + a ₆ + a ₅	a ₅ + a ₆	a ₄
4	a ₃		a ₃ + a ₅ + a ₆	a ₃ + a ₄ + a ₅	a ₄ + a ₅ + a ₆	a ₃
5			0	a ₃ + a ₅ + a ₆	a ₃ + a ₄ + a ₅	a ₂ = a ₄ + a ₅ + a ₆
6		OFF	0	0	a ₃ + a ₅ + a ₆	a ₁ = a ₃ + a ₄ + a ₅
7			0	0	0	a ₀ = a ₃ + a ₅ + a ₆

b)

Fig. 5.16 Cyclic encoder with LFSR and internal \oplus : a) block scheme; b) operation table for $g(x) = x^3 + x + 1$ and $m = 4$.

From the operation table it can be seen that $r(x)$, the remainder of $x^k i(x)/g(x)$ is calculated by the LFSR at the end of $m = 4$ clocks, being downloaded from the register during the last $k = 3$ clocks, when the gate P is OFF.

The register characteristic matrix is given by (5.149):

$$\mathbf{T} = \begin{bmatrix} g_{k-1} & 1 & \dots & 0 \\ g_{k-2} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ g_1 & 0 & \dots & 1 \\ g_0 & 0 & \dots & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix},$$

\mathbf{U} is calculated according to (5.145), and the control matrix (5.152) will be:

$$\mathbf{H} = [\mathbf{U} \ \mathbf{T}\mathbf{U} \ \mathbf{T}^2\mathbf{U} \ \mathbf{T}^3\mathbf{U} \ \mathbf{T}^4\mathbf{U} \ \mathbf{T}^5\mathbf{U} \ \mathbf{T}^6\mathbf{U}] = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{H}\mathbf{v}^T = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} = 0 \Rightarrow \begin{cases} a_1 \oplus a_2 \oplus a_3 \oplus a_6 = 0 \\ a_0 \oplus a_1 \oplus a_2 \oplus a_5 = 0 \\ a_0 \oplus a_2 \oplus a_3 \oplus a_4 = 0 \end{cases}$$

Processing the last equations one can obtain the encoding relationships for a_0 , a_1 , and a_2 .

Error detection cyclic decoders

As shown in 5.8.3, relation (5.115), the error detection condition is that the syndrome $s(x)$ (the remainder of $r(x)/g(x)$) is non-zero. This state has to be underlined by the decoder. We have already seen, when encoding, that the LFSR divides $x^k i(x)$ to $g(x)$, so it can also be used for decoding. In both cases we saw that, at the end of the n clocks, the LFSR state is zero; it follows that, when detecting the error, it is enough to test the LFSR final state. If this state is non-zero, the error is detected (see Fig. 5.17).

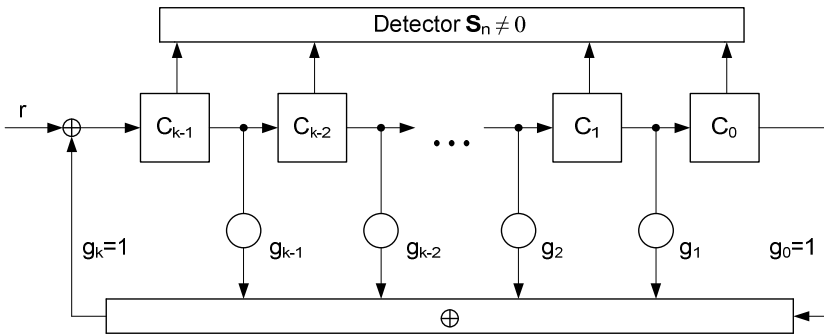


Fig. 5.17 Error detection cyclic decoder with LFSR and external \oplus

The syndrome \mathbf{S} , determined by the LFSR state at the moment n : $\mathbf{S}_n \neq 0$ is not the remainder of $r(x)/g(x)$, but a modified form of it which allows the error detection.

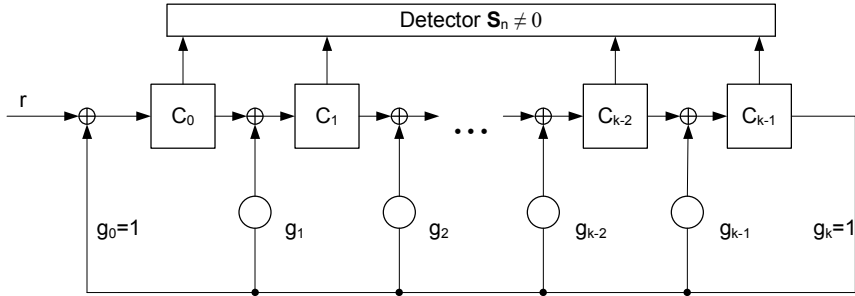


Fig. 5.18 Error detection cyclic decoder with LFSR and internal \oplus

For this decoder, the syndrome \mathbf{S} is calculated as the remainder of $r(x)/g(x)$ and is determined by the LFSR state at the moment n .

Example 5.15

Consider the cyclic code $H(7,4)$ with $g(x) = x^3 + x + 1$ and the received words: $\mathbf{r}_1 = [1\ 0\ 0\ 1\ 0\ 1\ 1]$ and $\mathbf{r}_2 = [0\ 0\ 0\ 1\ 0\ 1\ 1]$. Using a LFSR with external \oplus , check if the two words have been correctly received.

Solution

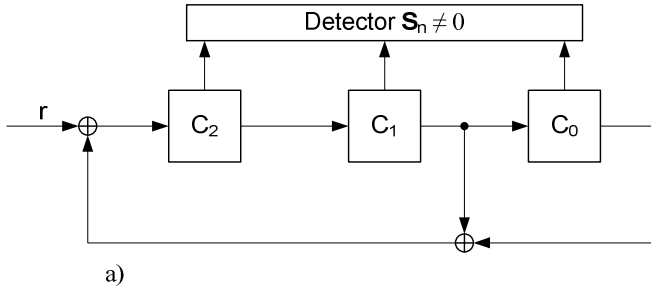
$$r_1(x) = x^6 + x^3 + x + 1$$

$$r_2(x) = x^3 + x + 1$$

It can be seen that $r_2(x)$ is the codeword $= g(x)$.

$$r_1(x)/g(x) = (x^6 + x^3 + x + 1)/(x^3 + x + 1) = \underbrace{x^3 + x + 1}_{q(x)} + \frac{x^2 + 1}{g(x)},$$

so $r(x) = x^2 + 1$; this means that $r_1(x)$ is erroneous. It can be emphasized also by the LFSR state ($\neq 0$) at the moment $n = 7$. The block scheme and the operation table of the decoder are given in Fig. 5.19:



a)

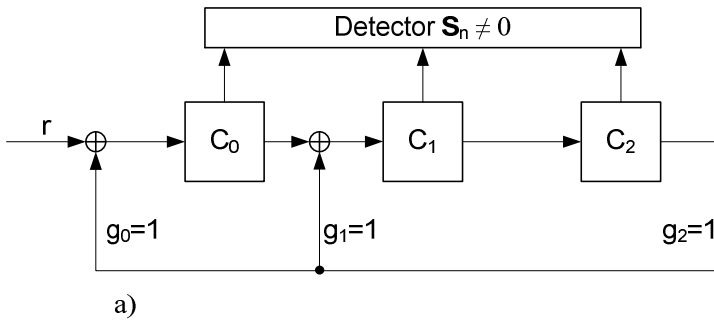
T		t_n		t_{n+1}					
		r		C ₂		C ₁		C ₀	
		1	2	1	2	1	2	1	2
1		1	0	1	0	0	0	0	0
2		0	0	0	0	1	0	0	0
3		0	0	1	0	0	0	1	0
4		1	1	0	1	1	0	0	0
5		0	0	1	0	0	1	1	0
6		1	1	0	0	1	0	0	1
7		1	1	0	0	0	0	1	0

b)

Fig. 5.19 a) Block scheme and b) the operating table of the cyclic decoder for $g(x) = x^3 + x + 1$

From the operating table we see that at the moment $n = 7$, for r_1 , the register state is $(001) = S_n \neq 0$; this state is not that one corresponding to the remainder $(1 0 1)$.

The block scheme and the operation table of a cyclic error detection decoder with LFSR and internal modulo two adders is given in Fig. 5.20.



T	t_n		t_{n+1}					
	r		C ₀		C ₁		C ₂	
	1	2	1	2	1	2	1	2
1	1	0	1	0	0	0	0	0
2	0	0	0	0	1	0	0	0
3	0	0	0	0	0	0	1	0
4	1	1	0	1	1	0	0	0
5	0	0	0	0	0	1	1	0
6	1	1	0	1	1	0	0	1
7	1	1	1	0	0	0	1	0

b)

Fig. 5.20 a) Block scheme and b) the operation table of a cyclic error detection decoder with LFSR and internal \oplus

Error Correction Cyclic Decoder (Meggitt Decoder)

Decoding for binary error correction must allow determining the erroneous position from the syndrome expression.

As we have already seen, error detection is possible at the end of the entire word reception, i.e. after n clocks. For binary codes correction it is necessary to know the position of the error. If it is possible to determine a fixed state of the syndrome register (SR) during $(n+1; 2n)$, when the erroneous bit is found in the last cell of an n -bit memory register (MR) in which the received word is serially loaded, the correction is immediately done summing modulo a “1” on the determined position. The correction is possible during $(n+1; 2n)$, therefore, the error correction with cyclic codes takes $2n$ clocks. One word length breaks are avoided during transmission using two identical decoders that operate in push-pull. The block scheme of an error correction cyclic decoder is given in Fig. 5.21.

The legend for Fig. 5.21 is:

- MR – memory register (n cells)
- LFSR (SR) – syndrome register (a LFSR identical with that one used for encoding)

- D – fix state detector: detects the SR unique state (fix state) when the erroneous bit is in the last cell (n) of the MR.
- C – correction cell (modulo two adder)
- P_1, P_2, P_1^*, P_2^* - gates ensuring the decoders push-pull operation.

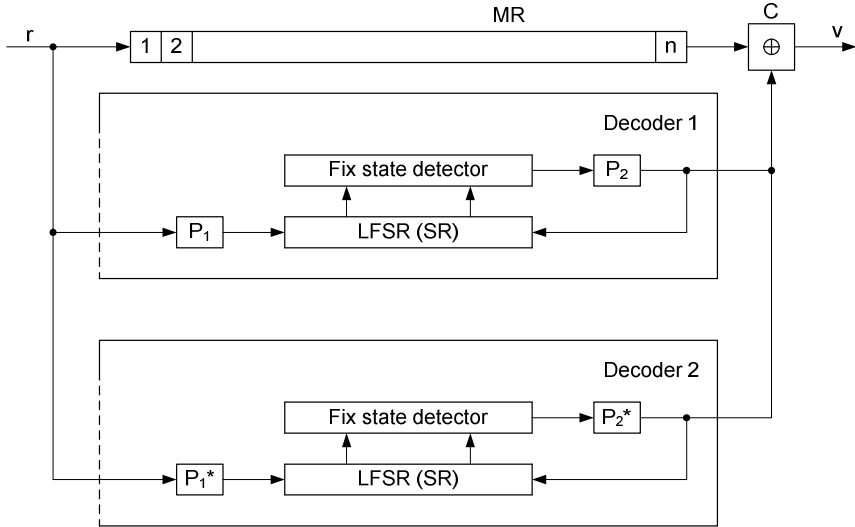


Fig. 5.21 General block scheme of an error correction cyclic decoder

The way this block scheme operates is described below:

- During $(1 \div n)$: $P_1 - ON$ $P_1^* - OFF$
 $P_2 - OFF$ $P_2^* - ON$

the received word enters the MR and simultaneously the first decoder.

- During $(n + 1 \div 2n)$: $P_1 - OFF$ $P_1^* - ON$
 $P_2 - ON$ $P_2^* - OFF$

it is possible to correct the received word during $(n + 1; 2n)$ in the correction cell C, detecting the SR fix state. The next word is received and simultaneously loaded in the second decoder and in the MR too, as this register becomes available when the previous word has been corrected too.

5.8.6 Cyclic One Error Correcting Hamming Code

The Hamming perfect code, studied in 5.4.5 can be cyclically encoded too. The relation (5.71) defining the perfect one error correcting code remains: $n = 2^k - 1$.

The differences between this code and the Hamming group code studied in 5.7.9 are the codeword structure and the encoding / decoding relations.

The structure of the cyclic Hamming code is:

$$\mathbf{v} = [\underbrace{a_0 a_1 \dots a_{k-1}}_{\substack{k \text{ control} \\ \text{symbols}}} \underbrace{a_k \dots a_{n-1}}_{\substack{m \text{ information} \\ \text{symbols}}}], \text{ (see relation (5.98))}$$

For encoding, a primitive generator polynomial $g(x)$ of degree k is chosen. The encoding can be done using LFSR, as shown in the Fig. 5.13, respectively Fig. 5.14. The encoding relations are obtained with:

$$\mathbf{H} \cdot \mathbf{v}^T = 0,$$

where $\mathbf{H} = [\mathbf{U} \ \mathbf{TU} \ \dots \ \mathbf{T}^{n-1}\mathbf{U}]$ (see relation (5.152)), with \mathbf{T} and \mathbf{U} determined as indicated in 5.8.5.

In what follows we will present in detail the error correction decoding procedures, determining that the syndrome register fix state is highlighted by the detector D, when the erroneous bit reaches the MR last cell of the decoder presented in Fig. 5.21.

Decoder with LFSR and external modulo 2 adders

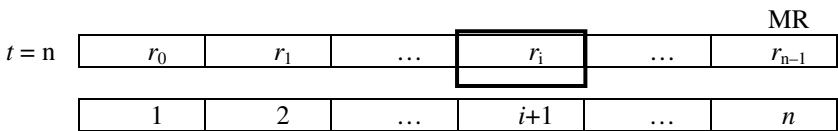
Suppose an error occurred on position r_i , $i = \overline{0, n-1}$ during transmission:

$$\mathbf{e} = [0 \dots e_i \dots 0]$$

At moment n , the syndrome \mathbf{S}_n is:

$$\mathbf{S} = \mathbf{Hr}^T = \mathbf{He}^T = [\mathbf{U} \ \mathbf{TU} \ \dots \ \mathbf{T}^{n-1}\mathbf{U}] \cdot \begin{bmatrix} 0 \\ \vdots \\ e_i \\ \vdots \\ 0 \end{bmatrix} = \mathbf{T}^i \mathbf{U} \tag{5.153}$$

At this moment (n), the erroneous bit is in the $(i+1)$ th cell of the MR:



The symbol r_i will reach the MR last cell after $(n - i - 1)$ clocks, therefore taking into account also the n clocks necessary to charge r in MR, at the moment $n + (n - i - 1)$. We are interested to find out the SR state at that moment. For this purpose, the SR will freely operate (without input, P_1 being blocked):

$$\begin{aligned}
 n: \quad \mathbf{S} &= \mathbf{T}^i \mathbf{U} \\
 n+1: \quad \mathbf{TS} &= \mathbf{T}^{i+1} \mathbf{U} \\
 &\vdots \\
 n+(n-i-1): \quad \mathbf{T}^{n-i-1} \mathbf{S} &= \mathbf{T}^{n-i-1} \mathbf{T}^i \mathbf{U} = \mathbf{T}^n \mathbf{T}^{-1} \mathbf{U} = \mathbf{IT}^{-1} \mathbf{U} = \mathbf{T}^{-1} \mathbf{U} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (5.154)
 \end{aligned}$$

\mathbf{T}^n being the k -th order unit matrix and $\mathbf{T}^{-1} \mathbf{U} = [1 \ 0 \dots 0]^T$, T indicating the transposition.

So, for a decoder with LFSR with external \oplus , the SR fix state is $[1 \ 0 \dots 0]$, meaning that all the k cells, except C_0 , will be $\mathbf{0}$. This state will be detected by D which will provide a '1', added in the correction cell C with r_i :

$$r_i + 1 = (a_i + 1) + 1 = a_i \quad (5.155)$$

implying the complementation of the bit value, so correcting r_i .

Remark

The occurrence of this state $[1 \ 0 \dots 0]$ during $(1, n)$ would cause false corrections, therefore it is necessary to block P_2 for all this period.

Decoder with LFSR and internal modulo two adders

For LFSR with internal \oplus the state of the LFSR indicates precisely the remainder, meaning that when the erroneous symbol reaches the last cell of the MR, the error word $e(x)$ corresponds to x^{n-1} , meaning that the state of the SR is given by:

$$\text{rem} \frac{x^{n-1}}{g(x)} = \text{SR fix state} \quad (5.156)$$

Example 5.16

Consider the cyclic one error correcting Hamming code $H(7,4)$ with $g(x) = x^3 + x + 1$; its encoding was analysed in example 5.15.

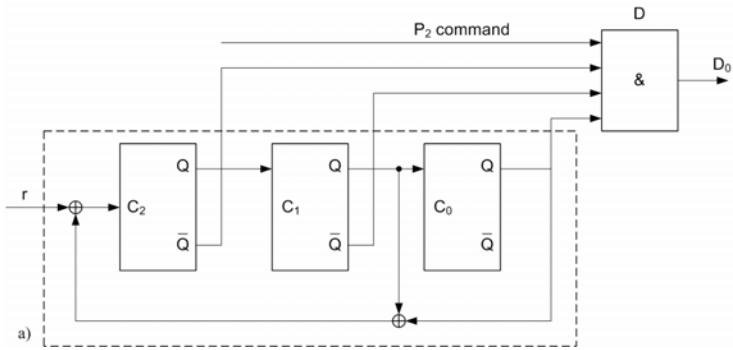
We choose the following transmitted word:

$$\mathbf{v} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \end{bmatrix}$$

and assume that a_4 ($i = 4$) is erroneous, so the reception is:

$$\mathbf{r} = [0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1]$$

We will illustrate for this r the error correction procedure using LFSR (figure 5.22).



Ck	SR			MR							D ₀		
	r	C ₂	C ₁	C ₀	1	2	3	4	5	6		7	
1	0	0	0	0	0	-	-	-	-	-	-	-	-
2	0	0	0	0	0	0	-	-	-	-	-	-	-
3	1	1	0	0	1	0	0	-	-	-	-	-	-
4	1	1	1	0	1	1	0	0	-	-	-	-	-
5	0	1	1	1	0	1	1	0	0	-	-	-	-
6	1	1	1	1	1	0	1	1	0	0	-	-	-
7	1	1	1	1	1	1	0	1	1	0	0	-	-
8	-	0	1	1	-	1	1	0	1	1	0	-	0
9	-	0	0	1	-	-	1	1	0	1	1	-	1

b)

Fig. 5.22 a) SR block scheme and b) operation of the cyclic decoder from Fig. 5.21, for the cyclic Hamming code (7,4) with $g(x) = x^3 + x + 1$; LFSR with external \oplus

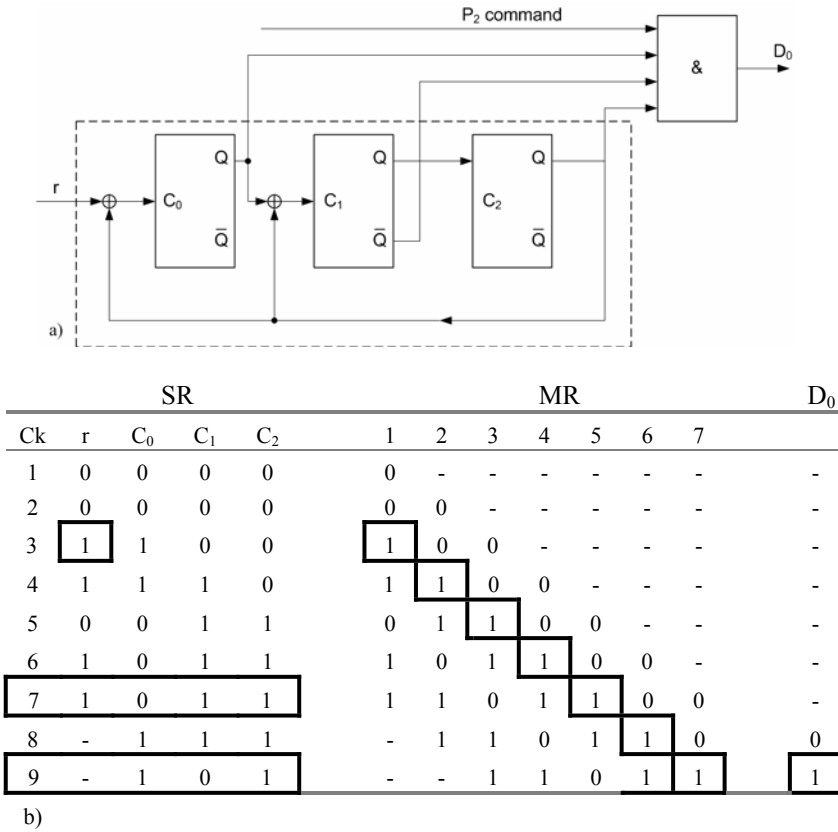


Fig. 5.23 a) SR block scheme and b) operation of the cyclic decoder from Fig. 5.21, for the cyclic Hamming code (7,4) with $g(x) = x^3 + x + 1$; LFSR with internal \oplus

5.8.7 Golay Code

The Golay codes are binary perfect codes, as Hamming codes.

The parameters of the Golay code are:

$$n = 23, m = 12, k = 11, t = 3 \tag{5.157}$$

Being a perfect code, the number of non-zero syndromes is equal to that of the correctable errors; therefore relation (5.47): $\sum_{i=1}^t C_n^i = 2^k - 1$ is satisfied for $t = 3$.

$$C_{23}^1 + C_{23}^2 + C_{23}^3 = 2^{11} - 1 = N \tag{5.158}$$

It results that the codeword roots are primitive elements α of $GF(2^{11})$ generated by a primitive polynomial $p(x)$ of degree $k = 11$ for which:

$$\alpha^N = 1 \tag{5.159}$$

Due to the fact that the codeword length is $n = 23$, the generator polynomial of degree 11 must have the same period $n = 23$, so all its α roots must be of the same order 23.

From $N = 2^{11} - 1$ elements of $\text{GF}(2^{11})$, we will choose a root α^i to be a root of:

$$X^n + 1 = 0 \quad (5.160)$$

and at the same time root of $g(x)$. Replacing α^i in (5.160) and taking into account (5.159), we obtain:

$$\alpha^{i \cdot n} = \alpha^N = 1 \quad (5.161)$$

from which results:

$$i = \frac{N}{n} = \frac{2^{11} - 1}{23} = 89 \quad (5.162)$$

In order to obtain a minimum number of control symbols, $g(x)$ must be a minimal polynomial of $\beta = \alpha^{89}$. Depending on how we choose $p(x)$ as generator for $\text{GF}(2^{11})$, we obtain [43], [54]:

$$g_1(x) = 1 + x^2 + x^4 + x^5 + x^6 + x^{10} + x^{11} \quad (5.163)$$

or:

$$g_2(x) = 1 + x + x^5 + x^6 + x^7 + x^9 + x^{11} \quad (5.164)$$

It is obvious that the two polynomials are dividers of $x^{23} + 1$:

$$x^{23} + 1 = (1 + x)g_1(x)g_2(x) \quad (5.165)$$

Decoding can be done using the correspondence table, (the syndrome based decoding table), or applying:

$$\mathbf{H} \cdot \mathbf{v}^T = \mathbf{0}$$

where

$$\mathbf{H} = \left[\beta^0 \ \beta^1 \ \dots \ \beta^{n-1} \right] \quad (5.166)$$

Elements β^i are represented by the table of modulo $p(x)$ residue, $p(x)$ polynomial of degree 11, and each element being an 11 bits matrix.

Observing the generator polynomials $g_1(x)$ and $g_2(x)$, it can be seen that the non-zero terms is 7, so according (5.110), the code distance is:

$$d = 2t + 1 \Rightarrow t = 3$$

It follows that this code can correct maximum 3 errors.

5.8.8 Fire Codes

Fire codes (1959) are the first error correcting cyclic codes used for burst error correction [28], [43].

An error correcting Fire code for bursts of length p is generated by the polynomial:

$$g(x) = (x^{2p-1} + 1)g_1(x) \quad (5.167)$$

where $g_1(x)$ is a primitive polynomial of degree m over $GF(2)$. By n_m we denote the period of $g_1(x)$. The length n of the code is:

$$n = \text{smallest common multiplier } \{2p-1, n_p\} \quad (5.168)$$

The number of control symbols is:

$$k = m + 2p - 1 \quad (5.169)$$

Fire encoding is done using LFSRs with external or internal modulo two adders.

Let us consider a burst error of length p :

$$e_1 = [\dots e_{n-t-p+1} \dots \varepsilon_{n-i} \dots \varepsilon_{n-t-1} e_{n-t} \dots] \quad (5.170)$$

or, written polynomially:

$$e_1(x) = [e_{n-t-p+1}x^{n-t-p+1} + \dots + \varepsilon_{n-i}x^{n-i} + \dots + \varepsilon_{n-t-1}x^{n-t-1} + e_{n-t}x^{n-t}] \quad (5.170.a)$$

where $e_i = 1$ represents the error position i . ε_j can be an erroneous position ($\varepsilon_j = 1$) or an error free position ($\varepsilon_j = 0$).

In order to include the case $p = 1$, when the burst has one error, we put:

$$e_{n-t-p+1} = \varepsilon_{n-t-p+1} = \begin{cases} 0, & \text{for } p = 1 \\ 1, & \text{for } p > 1 \end{cases} \quad (5.171)$$

In this case the error burst is:

$$\begin{aligned} e_1(x) &= e_{n-t-p+1}x^{n-t-p+1} + \dots + \varepsilon_{n-i}x^{n-i} + \dots + \varepsilon_{n-t-1}x^{n-t-1} + x^{n-t} = \\ &= x^{n-t-p+1}(\varepsilon_{n-t-p+1} + \dots + \varepsilon_{n-t-1}x^{p-2} + x^{p-1}) \end{aligned} \quad (5.172)$$

The syndrome is:

$$\begin{aligned}
 \mathbf{S} &= e_{n-t-p+1} \mathbf{T}^{n-t-p+1} \mathbf{U} + \dots + e_{n-i} \mathbf{T}^{n-i} \mathbf{U} + \mathbf{T}^{n-t} \mathbf{U} = \\
 &= \mathbf{T}^{-t} (e_{n-t-p+1} \mathbf{T}^{-p+1} + \dots + e_{n-t-1} \mathbf{T}^{-1} + \mathbf{I}) \mathbf{U} = \\
 &= \mathbf{T}^{-t} (\mathbf{I} + \sum_{j=1}^{l-1} e_{n-t-j} \mathbf{T}^{-j}) \mathbf{U}
 \end{aligned} \tag{5.173}$$

where \mathbf{T} is the LFSR characteristic matrix used for encoding, and \mathbf{U} the SR input matrix (both depending on the type of LFSR).

When the first erroneous symbol r_{n-1} reaches the MR last cell, which is after $t-1$ clocks, the SR state is:

$$\mathbf{T}^{t-1} \mathbf{S} = \mathbf{T}^{-1} (\mathbf{I} + \sum_{j=1}^{l-1} \varepsilon_{n-t-j} \mathbf{T}^{-j}) \mathbf{U} \tag{5.174}$$

Example 5.17

Design a Fire error correcting code for burst errors of length $p = 3$ and determine the block schemes for the encoding and decoding units with LFSR and external modulo two adders.

Solution

We choose a primitive polynomial, $g_1(x)$, of degree $m = 5$:

$$g_1(x) = 1 + x^2 + x^5$$

The code generator polynomial, according to (5.167), will be:

$$g(x) = (x^{2p-1} + 1)g_1(x) = (x^5 + 1)(1 + x^2 + x^5) = 1 + x^2 + x^7 + x^{10}$$

Polynomial $g_1(x)$ is a primitive one; it follows that its period is maximum and equals to:

$$n = 2^m - 1 = 2^5 - 1 = 31$$

From (5.168), the code length n is:

$$n = \text{smallest common multiplier } \{5; 31\} = 5 \cdot 31 = 155$$

The control symbols number being (5.169):

$$k = m + 2p - 1 = 5 + 5 = 10,$$

it follows that 145 is the number of information symbols.

The block schemes of Fire encoder and decoder implemented with LFSR and external adders are:

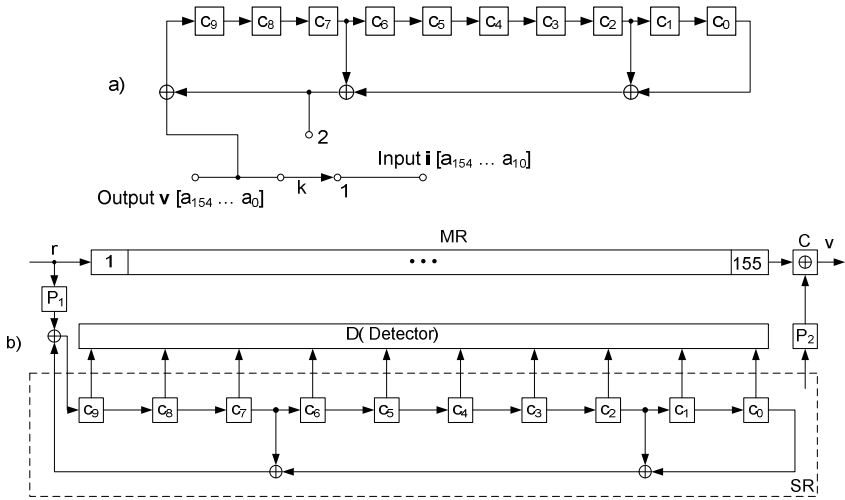


Fig. 5.24 Block schemes for Fire code with $g(x) = x^{10} + x^7 + x^2 + 1$ and $p = 3$: a) encoder and b) decoder

We will analyze the expression of the syndrome, given by (5.174), for bursts of lengths $p = 1$, $p = 2$ and $p = 3$.

- $p = 1$

The syndrome register state when the error is in the MR last cell (155) is:

$$\mathbf{T}^{-1}\mathbf{U} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & \dots & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

Therefore, the detector D detects the state 1 of the cell C_0 and the 0 states of all the other cells.

- $p = 2$

The SR state that must be detected when the first erroneous symbol of the burst is in the last MR cell:

$$\mathbf{T}^{-1}\mathbf{U} + \mathbf{T}^{-2}\mathbf{U} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

so the detector D detects the states 1 of the cells C_0 and C_1 , all the other cells being in 0.

- $p = 3$

The SR state that must be detected is:

$$\mathbf{T}^{-1}\mathbf{U} + \varepsilon\mathbf{T}^{-2}\mathbf{U} + \mathbf{T}^{-3}\mathbf{U} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \varepsilon \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \varepsilon \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

which means that for a burst as 1 0 1, the symbol $\varepsilon = 0$, so the state detected by the SR is:

$$[0 \ 0 \ 1 \ 0 \ \dots \ 0]^T$$

and for the burst 1 1 1, $\varepsilon = 0$, and the detected state is:

$$[0 \ 1 \ 1 \ 0 \ \dots \ 0]^T.$$

5.8.9 Reed–Solomon Codes

Reed-Solomon (RS) codes, proposed in 1960, are a variety of non-binary cyclic codes (the bits are replaced with characters), extremely convenient in many character oriented applications; they are used for independent and burst errors correction.

A RS word (block) of length n can be expressed as a vector:

$$\mathbf{v} = (\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{n-1}) \tag{5.175}$$

or as a polynomial:

$$v(x) = \mathbf{v}_{n-1}x^{n-1} + \dots + \mathbf{v}_1x + \mathbf{v}_0 \tag{5.176}$$

where each character \mathbf{v}_i is an element of Galois field $GF(q = 2^k)$, so it can be expressed with k bits.

We can associate decimals to $GF(2^k)$ elements, as follows:

$$GF(2^k) = \begin{pmatrix} 0 & 1 & \alpha & \alpha^2 & \dots & \alpha^{2^k-2} \\ 0 & 1 & 2 & 3 & \dots & n-1 \end{pmatrix} \tag{5.177}$$

where α is a primitive element of the $GF(2^k)$ field.

The parameters of a t errors correcting RS code are:

- codeword length n given by:

$$n = 2^k - 1 \tag{5.178}$$

where k represents the binary Galois field extension: $GF(2^k)$.

We have:

$$n = m + k' \quad (5.179)$$

where m is the information characters number and k' the control characters number.

- control characters number (k') can be determined with:

$$k' = 2t \quad (5.180)$$

and the code distance is: $d = 2t + 1$.

Relation (5.180) indicates the high efficiency of these codes, compared to the binary BCH codes; at the same relative redundancy the correction capacity is higher.

Example 5.18

The BCH code (15,7) can correct $t = k'/k = 8/2 = 4$ errors (see relation (5.110)).

According to (5.180), the RS code (15,7) can correct $t = k'/k = 8/2 = 4$ erroneous characters, at the same relative redundancy $k'/n = 8/15$.

RS encoding and decoding are very much similar to the binary BCH ones, the last one being a particular case of RS codes for $q = 2$.

We will analyse RS encoding and decoding both in time and in frequency [6].

Encoding

- *Time encoding*

The t errors correcting RS codes, as well as the BCH codes, are based on the existence of a generator polynomial, defined as the smallest common multiple associated to a number of $2t$ consecutive elements of the field $GF(2^k)$.

The encoding algorithm is based on the fact that $v(x)$ is divisible with $g(x)$, so the two polynomials have the same $2t$ roots:

$$g(x) = (x + \alpha^p)(x + \alpha^{p+1}) \dots (x + \alpha^{p+2t-1}) \quad (5.181)$$

where p is an arbitrary integer, usually 0 or 1.

The encoding process, as for all cyclic codes, can lead to a systematic or non-systematic code. Systematic encoding results in obtaining code words $v(x)$, in which the information symbols are on the first m most significant positions and the control symbols are on the last k' positions.

Example 5.19

Design a RS code having length $n = 7$, error correcting capacity $t = 2$ and systematically encode an arbitrary information sequence.

Solution

$GF(2^k)$ is determined with (5.178):

$$n = 2^k - 1 \Rightarrow k = 3 \Rightarrow GF(2^3)$$

The representation of the $GF(2^3)$ field is given in table found in Appendix A.10. The number of control characters using (5.180) is:

$$k' = 2t = 4$$

It follows RS(7,3), where 3 represents the information characters number (m). The generator polynomial is determined using relationship (5.184):

$$\begin{aligned} g(x) &= (x + \alpha^1)(x + \alpha^2)(x + \alpha^3)(x + \alpha^4) = x^4 + \alpha x^3 + x^2 + \alpha x + \alpha^3 = \\ &= x^4 + 4x^3 + x^2 + 2x + 4 \end{aligned}$$

In order to perform the encoding we choose an information sequence:

$$\mathbf{i} = \begin{pmatrix} 3 & 5 & 1 \\ \text{MSC} \end{pmatrix}$$

with polynomial representation:

$$i(x) = 3x^2 + 5x + 1 = \alpha^2 x^2 + \alpha^4 x + \mathbf{1} .$$

Using (5.98) we determine the systematic codeword:

$$\begin{aligned} x^{k'} i(x) &= x^{2t} i(x) = x^4 (\alpha^2 x^2 + \alpha^4 x + \mathbf{1}) = \alpha^2 x^6 + \alpha^4 x^5 + x^4 \\ x^{2t} i(x) / g(x) &= (\alpha^2 x^6 + \alpha^4 x^5 + x^4) / (x^4 + \alpha^3 x^3 + x^2 + \alpha x + \alpha^3) = \\ &= \alpha^2 x^2 + x + \alpha^4 + (\alpha^3 x^3 + \alpha^3 x^2 + \alpha^2 x + \mathbf{1}) / g(x) \\ v(x) &= \underbrace{\alpha^2 x^6 + \alpha^4 x^5 + x^4}_{x^{2t} i(x)} + \underbrace{\alpha^3 x^3 + \alpha^3 x^2 + \alpha^2 x + \mathbf{1}}_{\text{remaind } \frac{x^{2t} i(x)}{g(x)}} = q(x)g(x) \\ v &= \begin{pmatrix} \alpha^2 & \alpha^4 & \mathbf{1} & \alpha^3 & \alpha^3 & \alpha^2 & \mathbf{1} \\ \text{MSC} \end{pmatrix} = (3 \ 5 \ 1 \ 4 \ 4 \ 3 \ 1) \end{aligned}$$

Remark

Tables for RS generator polynomials having different lengths n and t can be found in Appendix A.12.

• Frequency encoding [6]

All the frequency expressions are obtained using the Fourier transform in the Galois field. The frequency domain offers, in certain situations (especially when decoding), a series of advantages: easier computing and simpler implementation (we use fast computation algorithms for the Fourier transform and also digital signal processors DSPs).

The discrete Fourier transform (DFT) of \mathbf{v} is a vector \mathbf{V} of length n with symbols $V_k \in GF(2^k)$ given by:

$$V_k = \sum_{i=0}^{n-1} \alpha^{ik} v_i \quad (5.182)$$

where α is a primitive element of $GF(2^k)$.

The polynomial associated to an n length word is, in frequency:

$$\mathbf{V}(x) = \sum_{k=0}^{n-1} \mathbf{V}_k x^k \quad (5.183)$$

The Reverse Discrete Fourier Transform (RDFT) is defined as follows:

$$\mathbf{v}_i = \frac{1}{n} \sum_{k=0}^{n-1} \mathbf{a}^{-ik} \mathbf{V}_k \quad (5.184)$$

so, taking into consideration (5.183) we can write:

$$\mathbf{v}_i = \frac{1}{n} \mathbf{V}(\mathbf{a}^{-i}) \quad (5.185)$$

$g(x)$ must divide the codeword, so they have the same roots:

$$\begin{aligned} \mathbf{v}(\mathbf{a}^p) &= \mathbf{v}_0 + \mathbf{v}_1 \mathbf{a} + \dots + \mathbf{v}_{n-1} \mathbf{a}^{n-1} = \mathbf{0} \\ \dots & \\ \mathbf{v}(\mathbf{a}^{p+2t-1}) &= \mathbf{v}_0 + \mathbf{v}_1 \mathbf{a}^{p+2t-1} + \dots + \mathbf{v}_{n-1} \mathbf{a}^{(p+2t-1)(n-1)} = \mathbf{V}_{p+2t-1} = \mathbf{0} \end{aligned} \quad (5.186)$$

It follows that the first $2t$ components of \mathbf{V} are zero.

Example 5.20

For RS(7,3) from example 5.19 we determined:

$$\mathbf{v}(x) = \mathbf{a}^6 x^6 + \mathbf{a}^4 x^5 + x^4 + \mathbf{a}^3 x^3 + \mathbf{a}^3 x^2 + \mathbf{a}^2 x + \mathbf{1}$$

The frequency expressions of the n components are:

$$\begin{aligned} \mathbf{V}_1 = \mathbf{v}(\mathbf{a}^1) &= \mathbf{a}^8 + \mathbf{a}^9 + \mathbf{a}^4 + \mathbf{a}^6 + \mathbf{a}^5 + \mathbf{a}^3 + \mathbf{1} = \mathbf{0} \\ \mathbf{V}_2 = \mathbf{v}(\mathbf{a}^2) &= \mathbf{a}^{14} + \mathbf{a}^{14} + \mathbf{a}^4 + \mathbf{a}^9 + \mathbf{a}^7 + \mathbf{a}^3 + \mathbf{1} = \mathbf{0} \\ \mathbf{V}_3 = \mathbf{v}(\mathbf{a}^3) &= \mathbf{a}^{20} + \mathbf{a}^{19} + \mathbf{a}^{12} + \mathbf{a}^{12} + \mathbf{a}^9 + \mathbf{a}^5 + \mathbf{1} = \mathbf{0} \\ \mathbf{V}_4 = \mathbf{v}(\mathbf{a}^4) &= \mathbf{a}^{26} + \mathbf{a}^{24} + \mathbf{a}^{16} + \mathbf{a}^{15} + \mathbf{a}^{11} + \mathbf{a}^6 + \mathbf{1} = \mathbf{0} \\ \mathbf{V}_5 = \mathbf{v}(\mathbf{a}^5) &= \mathbf{a}^{32} + \mathbf{a}^{29} + \mathbf{a}^{20} + \mathbf{a}^{18} + \mathbf{a}^{13} + \mathbf{a}^7 + \mathbf{1} = \mathbf{a} \\ \mathbf{V}_6 = \mathbf{v}(\mathbf{a}^6) &= \mathbf{a}^{38} + \mathbf{a}^{34} + \mathbf{a}^{24} + \mathbf{a}^{21} + \mathbf{a}^{15} + \mathbf{a}^8 + \mathbf{1} = \mathbf{a}^6 \\ \mathbf{V}_7 = \mathbf{v}(\mathbf{a}^7) &= \mathbf{a}^{44} + \mathbf{a}^{39} + \mathbf{a}^{28} + \mathbf{a}^{24} + \mathbf{a}^{17} + \mathbf{a}^9 + \mathbf{1} = \mathbf{a}^6 \end{aligned}$$

Algebraic decoding for RS codes

For RS codes, besides the knowledge of error position given by the *locators* \mathbf{X}_k (sufficient for BCH decoding) it is also necessary to determine the *error value* \mathbf{Y}_k . The error value, added to the erroneous value allows its correction.

In what follows we will present two of the most efficient decoding algorithms: *Peterson with Chien search* and *Berlekamp*. The first algorithm will be presented in time domain, while the second in frequency.

Peterson algorithm with Chien search was described at BCH decoding. However there are some differences between RS and BCH decoding algorithms, and all these differences will be emphasised step by step.

1. Error syndrome calculation:

$$\mathbf{S} = (\mathbf{S}_1, \mathbf{S}_2, \dots, \mathbf{S}_{2t})$$

$$\mathbf{S}_i = r(\alpha^i) = \sum_{k=1}^t \mathbf{Y}_k \mathbf{X}_k^i; \quad 1 \leq i \leq 2t \tag{5.187}$$

2. Determination of σ_t coefficients of the error polynomial $\sigma(x)$ as functions of the syndromes \mathbf{S}_j calculated to 1:

$$\mathbf{S}_{t+i} + \sigma_1 \mathbf{S}_{t+i-1} + \dots + \sigma_t \mathbf{S}_i = \mathbf{0}, \quad i = \overline{1, t}$$

The coefficients table $\sigma_t = f(\mathbf{S}_i)$ valid for BCH codes can not be used for RS codes because the equality $\alpha_{2k} = \alpha_k^2$ is not valid any more. We give a coefficients table for $t=1$ and $t=2$, and the reader can easily determine the coefficients α_t for $t > 2$.

Table 5.12 σ_t coefficients of the error polynomial for RS codes

t	σ_i
1	$\sigma_1 = \mathbf{S}_1 / \mathbf{S}_2$
2	$\sigma_1 = (\mathbf{S}_1 \mathbf{S}_4 + \mathbf{S}_3 \mathbf{S}_2) / (\mathbf{S}_2^2 + \mathbf{S}_1 \mathbf{S}_3)$ $\sigma_2 = (\mathbf{S}_2 \mathbf{S}_4 + \mathbf{S}_3^2) / (\mathbf{S}_2^2 + \mathbf{S}_1 \mathbf{S}_3)$

3. Determination of locators using Chien search, identical for BCH and RS codes:

$$\sum_{i=1}^t \sigma_i \alpha^{ij} = 1, \quad j = \overline{1, n} \quad \mathbf{X}_k = \alpha^{n-j}$$

4. Determination of \mathbf{Y}_k – the error values

The value \mathbf{Y}_k is found starting from (5.187):

$$\mathbf{S}_i = r(\alpha^i) = \sum_{k=1}^t \mathbf{Y}_k \mathbf{X}_k^i$$

where the locators \mathbf{X}_k were determined at 3 and the syndromes \mathbf{S}_i calculated at 1.

This relation represents a linear system of t equations with t unknowns \mathbf{Y}_k :

$$\begin{cases} \mathbf{Y}_1\mathbf{X}_1^1 + \mathbf{Y}_2\mathbf{X}_2^1 + \dots + \mathbf{Y}_t\mathbf{X}_t^1 = \mathbf{S}_1 \\ \mathbf{Y}_1\mathbf{X}_1^2 + \mathbf{Y}_2\mathbf{X}_2^2 + \dots + \mathbf{Y}_t\mathbf{X}_t^2 = \mathbf{S}_2 \\ \dots \\ \mathbf{Y}_1\mathbf{X}_1^t + \mathbf{Y}_2\mathbf{X}_2^t + \dots + \mathbf{Y}_t\mathbf{X}_t^t = \mathbf{S}_t \end{cases} \quad (5.188)$$

Using Cramer rule, the solution is determined as follows:

$$\mathbf{Y} = \begin{pmatrix} \frac{\mathbf{D}'_1}{\mathbf{D}'} & \frac{\mathbf{D}'_2}{\mathbf{D}'} & \dots & \frac{\mathbf{D}'_t}{\mathbf{D}'} \end{pmatrix} \quad (5.189)$$

in which:

$$\mathbf{D}' = \begin{vmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \dots & \mathbf{X}_t \\ \mathbf{X}_1^2 & \mathbf{X}_2^2 & \dots & \mathbf{X}_t^2 \\ \dots & \dots & \dots & \dots \\ \mathbf{X}_1^t & \mathbf{X}_2^t & \dots & \mathbf{X}_t^t \end{vmatrix} \quad (5.190)$$

and $\mathbf{D}'_j, j = \overline{1, t}$ are characteristic determinants of the system (5.188).

The expressions for the coefficients \mathbf{Y}_k are given in table 5.13, for $k = 1$ and $k = 2$; the table can be easily completed for $k > 2$ using (5.188) to (5.190).

Table 5.13 \mathbf{Y}_k coefficients for RS codes ($t=1,2$)

t	\mathbf{Y}_k
1	$\mathbf{Y}_1 = \frac{\mathbf{S}_1^2}{\mathbf{S}_2}$
2	$\mathbf{Y}_1 = \frac{\mathbf{S}_1\mathbf{X}_2 + \mathbf{S}_2}{\mathbf{X}_1\mathbf{X}_2 + \mathbf{X}_1^2}$
	$\mathbf{Y}_2 = \frac{\mathbf{S}_1\mathbf{X}_1 + \mathbf{S}_2}{\mathbf{X}_1\mathbf{X}_2 + \mathbf{X}_2^2}$

5. Error correction

Correcting the character \mathbf{r}_{n-j} whose position \mathbf{X}_k was determined at step 3 is performed with:

$$\mathbf{v}_{n-j} = \mathbf{r}_{n-j} \oplus \mathbf{Y}_k \quad (5.191)$$

Example 5.21

Let us consider RS(7,3) with $g(x)$ determined in example 5.19 and the received word:

$$\mathbf{r} = \begin{pmatrix} 0 & 5 & 2 & 4 & 4 & 3 & 1 \\ \text{MSC} \end{pmatrix}$$

Determine whether \mathbf{r} is erroneous or not. If errors do exist, apply Peterson algorithm with Chien search and determine the correct word.

Solution

$$\mathbf{r}(x) = \alpha^4 x^5 + \alpha x^4 + \alpha^3 x^3 + \alpha^3 x^2 + \alpha^2 x + 1$$

1.

$$\mathbf{S}_1 = \mathbf{r}(\alpha) = \alpha^3$$

$$\mathbf{S}_2 = \mathbf{r}(\alpha^2) = \alpha^5$$

$$\mathbf{S}_3 = \mathbf{r}(\alpha^3) = \alpha^5$$

$$\mathbf{S}_4 = \mathbf{r}(\alpha^4) = 0 \Rightarrow \mathbf{S} = (\alpha^3 \ \alpha^5 \ \alpha^5 \ 0)$$

$$2. \quad \sigma_1 = (\mathbf{S}_1 \mathbf{S}_4 + \mathbf{S}_3 \mathbf{S}_2) / (\mathbf{S}_2^2 + \mathbf{S}_1 \mathbf{S}_3) = \alpha^5 \alpha^5 / (\alpha^{10} + \alpha^3 \alpha^5) = \alpha^3$$

$$\sigma_2 = (\mathbf{S}_2 \mathbf{S}_4 + \mathbf{S}_3^2) / (\mathbf{S}_2^2 + \mathbf{S}_1 \mathbf{S}_3) = \alpha^{10} / (\alpha^{10} + \alpha^3 \alpha^5) = \alpha^3$$

$$3. \quad j=1 \quad \sigma_1 \alpha^{11} + \sigma_2 \alpha^{21} = 1 \quad \text{so position } n-j = 7-1 = 6 \text{ is erroneous: } \mathbf{X}_1 = \alpha^6$$

$$j=2 \quad \sigma_1 \alpha^{12} + \sigma_2 \alpha^{22} = \alpha^2$$

$$j=3 \quad \sigma_1 \alpha^{13} + \sigma_2 \alpha^{23} = 1 \quad \text{so position } n-j = 7-3 = 4 \text{ is erroneous: } \mathbf{X}_1 = \alpha^4$$

$$j=4 \quad \sigma_1 \alpha^{14} + \sigma_2 \alpha^{24} \neq 1$$

$$j=5 \quad \sigma_1 \alpha^{15} + \sigma_2 \alpha^{25} \neq 1$$

$$j=6 \quad \sigma_1 \alpha^{16} + \sigma_2 \alpha^{26} \neq 1$$

$$j=7 \quad \sigma_1 \alpha^{17} + \sigma_2 \alpha^{27} \neq 1$$

$$4. \quad \mathbf{Y}_1 = \frac{\mathbf{S}_1 \mathbf{X}_2 + \mathbf{S}_2}{\mathbf{X}_1 \mathbf{X}_2 + \mathbf{X}_2^2} = \alpha^2$$

$$\mathbf{Y}_2 = \frac{\mathbf{S}_1 \mathbf{X}_1 + \mathbf{S}_2}{\mathbf{X}_1 \mathbf{X}_2 + \mathbf{X}_2^2} = \alpha^2$$

$$5. \quad \mathbf{v}_6 = \mathbf{r}_6 \oplus \mathbf{Y}_1 = \mathbf{0} \oplus \alpha^2 = 3$$

$$\mathbf{v}_4 = \mathbf{r}_4 \oplus \mathbf{Y}_2 = \alpha \oplus \alpha^3 = 1$$

The correct word is $\mathbf{v} = (3 \ 5 \ 1 \ 4 \ 4 \ 3 \ 1)$.

Berlekamp – Massey algorithm

Improvements of Peterson algorithm were brought by Berlekamp and Massey [6]. The algorithm can be implemented in time as well as in frequency, being preferred the frequency approach for powerful correcting codes ($t > 5$), because of its high processing speed. As we will see later on, in the frequency domain we deal with vectors having the dimension $2t$, whereas in time the vectors dimension is n ; n is usually much larger than $2t$, resulting a higher processing volume and subsequently, a longer decoding time.

In what follows we give a brief presentation of Berlekamp algorithm in frequency omitting the demonstrations, which can be found in [6].

Let us consider a received word affected by t additive errors:

$$\mathbf{r}_i = \mathbf{v}_i + \mathbf{e}_i; 0 \leq i \leq n-1 \quad (5.192)$$

where $\mathbf{e}_i \neq \mathbf{0}$ on the t erroneous positions and zero elsewhere.

Applying the DFT (see 5.182), we obtain:

$$\mathbf{R}_i = \mathbf{V}_i + \mathbf{E}_i; 0 \leq i \leq n-1 \quad (5.193)$$

As shown in (5.186), the first $2t$ components of the vector \mathbf{V} are zero, so:

$$\mathbf{S}_k = \mathbf{R}_k = \mathbf{E}_k; 0 \leq k \leq 2t-1 \quad (5.194)$$

This relation shows that, at receiver, one can easily determine the $2t$ components of the error word from the received word transposed in the frequency domain.

The idea used in Berlekamp algorithm is to generate the other $n-2t$ components of the error word from the free evolution of a feedback shift register initialized with the first $2t$ components of this error word.

The algorithm allows determining the error locator polynomial $\Lambda(x)$ which is exactly the polynomial of the LFSR initialized with the first $2t$ components \mathbf{S}_k :

$$\Lambda(x) =: \prod_{i=1}^t (x\alpha^i + 1) \quad (5.195)$$

If i is an erroneous position, than α^{-i} is a root of the locator polynomial:

$$\Lambda(\alpha^{-i}) = \mathbf{0} \quad (5.196)$$

According to (5.184), the time components of the error vector are:

$$\mathbf{e}_i = \frac{1}{n} \sum_{k=0}^{n-1} \alpha^{-i \cdot k} \mathbf{E}_k = E(\alpha^{-i}) \neq \mathbf{0} \quad (5.197)$$

If i is not an erroneous position, α^{-i} is not a root of the locator polynomial, so:

$$\Lambda(\alpha^{-i}) \neq \mathbf{0} \quad (5.198)$$

and the error value, in time, will be:

$$\mathbf{e}_i = E(\mathbf{a}^{-i}) = \mathbf{0} \quad (5.199)$$

It results:

$$\Lambda(\mathbf{a}^{-i})E(\mathbf{a}^{-i}) = \mathbf{0}; i = \overline{0, n-1} \quad (5.200)$$

so $\Lambda(x)E(x)$ must be a multiple of the polynomial:

$$x^n + 1 = \prod_{i=0}^n (x + \mathbf{a}^{-i}) \quad (5.201)$$

or:

$$\Lambda(x)E(x) = 0 \pmod{(x^n + 1)} \quad (5.202)$$

Starting from (5.202) we get the following equation system:

$$E_j = \sum_{k=1}^t \Lambda_k E_{j-k}, j = \overline{0, n-1} \quad (5.203)$$

The system (5.203) can be divided into two sub-systems:

$$\mathbf{E}_j = \sum_{k=1}^t \Lambda_k \mathbf{E}_{j-k}, j = \overline{0, 2t-1} \quad (5.203.a)$$

a system with t known coefficients \mathbf{E}_i and as unknowns the locator polynomial coefficients Λ_k , respectively:

$$\mathbf{E}_j = \sum_{k=1}^t \Lambda_k \mathbf{E}_{j-k}, j = \overline{2t, n-1} \quad (5.203.b)$$

The subsystem (5.203.b) determines the other $n-2t$ components of the error vector in the frequency domain, obtained by the free evolution of the feedback shift register.

The free evolution of this LFSR leads to extremely long computation time for large codeword lengths.

This can be overcome in the following way:

- re-write relation (5.202) as:

$$\Lambda(x)E(x) = \Gamma(x)(x^n + 1) \quad (5.204)$$

- computing (5.204) at \mathbf{a}^{-i} , index i expressing the erroneous position, we obtain a non-determination which can be removed by derivation:

$$\Lambda'(x)E(x) + \Lambda(x)E'(x) = \Gamma'(x)(x^n + 1) + nx^{n-1}\Gamma(x) + \Gamma'(x) \quad (5.205)$$

Replacing x with \mathbf{a}^{-i} , it results:

$$\Lambda'(\mathbf{a}^{-i})E(\mathbf{a}^{-i}) = n\mathbf{a}^i\Gamma(\mathbf{a}^{-i}) \quad (5.206)$$

and according to (5.184), we have:

$$\mathbf{e}_i = \frac{1}{n}E(\mathbf{a}^{-i}) = \frac{\mathbf{a}^i\Gamma(\mathbf{a}^{-i})}{\Lambda'(\mathbf{a}^{-i})} \quad (5.207)$$

These are exactly the error vector components in time domain.

The polynomials $\Lambda(x)$ and $\Gamma(x)$ are calculated applying Berlekamp and Massey theorem, which states the followings:

- Let us consider $v_0, v_1, \dots, v_{2t-1}$ given and $A(x)$, $B(x)$ two polynomials. For the following initial conditions:

$$\Lambda^{(0)}(x) = 1, B^{(0)}(x) = 1, \Gamma^{(0)}(x) = 0, A^{(0)}(x) = -x^{-1}, L_0 = 0, r = 0,$$

the next $2t$ iterations:

$$\begin{bmatrix} \Lambda^{(r)}(x) \\ B^{(r)}(x) \end{bmatrix} = \begin{bmatrix} 1 & \Delta x \\ \partial_r \Delta^{-1} & (1 - \partial_r)x \end{bmatrix} \begin{bmatrix} \Lambda^{(r-1)}(x) \\ B^{(r-1)}(x) \end{bmatrix} \quad (5.208)$$

$$\begin{bmatrix} \Gamma^{(r)}(x) \\ A^{(r)}(x) \end{bmatrix} = \begin{bmatrix} 1 & \Delta x \\ \partial_r \Delta^{-1} & (1 - \partial_r)x \end{bmatrix} \begin{bmatrix} \Gamma^{(r-1)}(x) \\ A^{(r-1)}(x) \end{bmatrix}$$

where

$$\partial_r = \begin{cases} 1, & \text{if } \Delta_r \neq 0, \text{ or } 2L_{r-1} \leq r-1 \\ 0, & \text{otherwise} \end{cases} \quad (5.209)$$

$$\Delta = \sum_{k=0}^r \Lambda_r \mathbf{E}_{r-1-k} \quad (5.210)$$

Δ is also known as the *discrepancy* between the calculated LFSR output and the known values of this output:

$$L_r = \max(L_{r-1}, r - L_{r-1}) \quad (5.211)$$

determine the two polynomials.

In other words, we have: $\Lambda^{(2t)}(x) = \Lambda(x)$ and $\Gamma^{(2t)}(x) = \Gamma(x)$.

Although we have already discussed the case in which $p = 0$, this algorithm can also be implemented in the general case, when $p \neq 0$, but with some minor changes:

- when initializing: $A^{(0)}(x) = -x^{-p-1}$
- when computing the discrepancy: $\Delta_r = \sum_{k=0}^r \Lambda_k E_{r-1-k+p}$.

The flow-chart of Berlekamp-Massey algorithm is presented in Fig. 5.25.

Example 5.22

We will decode the word \mathbf{r} from example 5.21, using Berlekamp-Massey algorithm.

$$\mathbf{r} = \begin{pmatrix} 0 & 5 & 2 & 4 & 4 & 3 & 1 \\ \text{MSC} \end{pmatrix}$$

1. Syndrome calculation: $S_i = E_i$, $i = \overline{1,2t}$

$$\begin{aligned} S_1 &= \mathbf{r}(\alpha) = \alpha^3 \\ S_2 &= \mathbf{r}(\alpha^2) = \alpha^5 \\ S_3 &= \mathbf{r}(\alpha^3) = \alpha^5 \\ S_4 &= \mathbf{r}(\alpha^4) = \mathbf{0} \\ \mathbf{S} &= (\alpha^3, \alpha^5, \alpha^5, \mathbf{0}) \end{aligned}$$

Initializing the algorithm:

$$\begin{aligned} \mathbf{E}_1 &= \mathbf{S}_1 = \alpha^3 \\ \mathbf{E}_2 &= \mathbf{S}_2 = \alpha^5 \\ \mathbf{E}_3 &= \mathbf{S}_3 = \alpha^5 \\ \mathbf{E}_4 &= \mathbf{S}_4 = \mathbf{0} \\ A(x) &= x^{p-1} = 1, \Lambda(x) = 1, \Gamma(x) = 0, B(x) = 1 \\ L &= 0, r = 0 \end{aligned}$$

Iteration $r = 1$

$$\begin{aligned} \Delta &= \Lambda_0 \mathbf{E}_1 = \alpha^3 \\ \Delta &\neq \mathbf{0}; L = r - L = 1 \\ \partial_1 &= 1 \\ \begin{bmatrix} \Lambda(x) \\ B(x) \end{bmatrix} &= \begin{bmatrix} \mathbf{1} & \alpha^3 x \\ \alpha^4 & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Lambda(x) \\ B(x) \end{bmatrix} = \begin{bmatrix} \mathbf{1} + \alpha^3 x \\ \alpha^4 \end{bmatrix} \\ \begin{bmatrix} \Gamma(x) \\ A(x) \end{bmatrix} &= \begin{bmatrix} \mathbf{1} & \alpha^3 x \\ \alpha^4 & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Lambda(x) \\ B(x) \end{bmatrix} = \begin{bmatrix} \alpha^3 x \\ \mathbf{0} \end{bmatrix} \end{aligned}$$

Iteration $r = 2$

$$\Delta = \Lambda_0 \mathbf{E}_2 + \Lambda_1 \mathbf{E}_1 = \mathbf{a}$$

$$\Delta \neq \mathbf{0}; 2L > r - 1 \Rightarrow$$

$$\partial_2 = 0$$

$$\begin{bmatrix} \Lambda(x) \\ \mathbf{B}(x) \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{a}x \\ \mathbf{0} & x \end{bmatrix} \begin{bmatrix} \Lambda(x) \\ \mathbf{B}(x) \end{bmatrix} = \begin{bmatrix} \mathbf{1} + \mathbf{a}^3 x + \mathbf{a}^5 x \\ \mathbf{a}^4 x \end{bmatrix} = \begin{bmatrix} \mathbf{1} + \mathbf{a}^2 x \\ \mathbf{a}^4 x \end{bmatrix}$$

$$\begin{bmatrix} \Gamma(x) \\ \mathbf{A}(x) \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{a}x \\ \mathbf{0} & x \end{bmatrix} \begin{bmatrix} \Lambda(x) \\ \mathbf{B}(x) \end{bmatrix} = \begin{bmatrix} \mathbf{a}^3 x \\ \mathbf{0} \end{bmatrix}$$

Iteration $r = 3$

$$\Delta = \Lambda_0 \mathbf{E}_3 + \Lambda_1 \mathbf{E}_2 = \mathbf{a}^4$$

$$\Delta \neq \mathbf{0}; 2L \leq r - 1 \Rightarrow$$

$$\partial_3 = 1, L = r - L = 2$$

$$\begin{bmatrix} \Lambda(x) \\ \mathbf{B}(x) \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{a}^4 x \\ \mathbf{a}^3 & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Lambda(x) \\ \mathbf{B}(x) \end{bmatrix} = \begin{bmatrix} \mathbf{1} + \mathbf{a}^3 x + \mathbf{a}x^2 \\ \mathbf{a}^3 + \mathbf{a}^5 x \end{bmatrix}$$

$$\begin{bmatrix} \Gamma(x) \\ \mathbf{A}(x) \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{a}^4 x \\ \mathbf{a}^3 & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Lambda(x) \\ \mathbf{B}(x) \end{bmatrix} = \begin{bmatrix} \mathbf{a}^3 x \\ \mathbf{a}^6 x \end{bmatrix}$$

Iteration $r = 4$

$$\Delta = \Lambda_1 \mathbf{E}_3 + \Lambda_2 \mathbf{E}_2 = \mathbf{a}^2$$

$$\Delta \neq \mathbf{0}; 2L > r - 1 \Rightarrow$$

$$\partial_3 = 0$$

$$\begin{bmatrix} \Lambda(x) \\ \mathbf{B}(x) \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{a}^2 x \\ \mathbf{0} & x \end{bmatrix} \begin{bmatrix} \Lambda(x) \\ \mathbf{B}(x) \end{bmatrix} = \begin{bmatrix} \mathbf{1} + \mathbf{a}^3 x + \mathbf{a}^3 x^2 \\ \mathbf{a}^3 x + \mathbf{a}^5 x^2 \end{bmatrix}$$

$$\begin{bmatrix} \Gamma(x) \\ \mathbf{A}(x) \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{a}^2 x \\ \mathbf{0} & x \end{bmatrix} \begin{bmatrix} \Lambda(x) \\ \mathbf{B}(x) \end{bmatrix} = \begin{bmatrix} \mathbf{a}^3 x + \mathbf{a}x^2 \\ \mathbf{a}^6 x^2 \end{bmatrix}$$

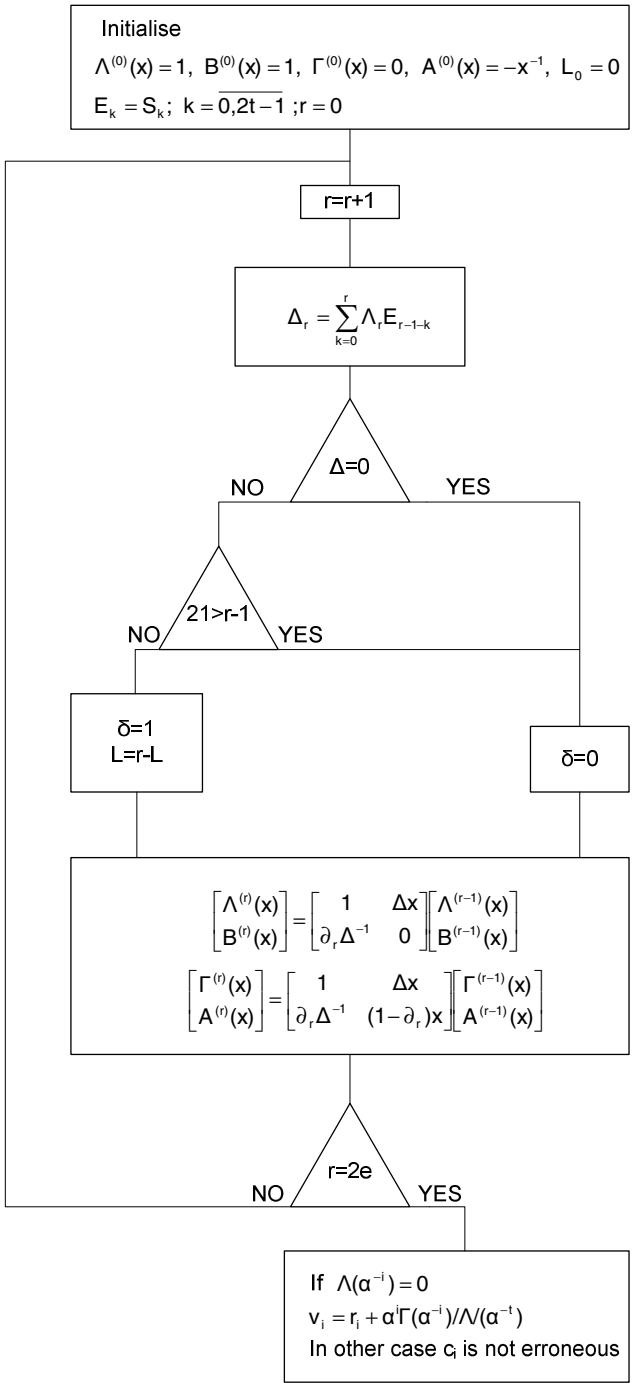


Fig. 5.25 Flow-chart corresponding to Berlekamp-Massey algorithm.

The two polynomials are:

$$\begin{aligned}\Lambda(x) &= \mathbf{1} + \alpha^3 x + \alpha^3 x^2, \\ \Gamma(x) &= \alpha^3 x + \alpha x^2, \\ \Lambda(\alpha^{-6}) &= \Lambda(\alpha) = \mathbf{1} + \alpha^4 + \alpha^5 = \mathbf{0} . \\ \Lambda(\alpha^{-5}) &\neq \mathbf{0} \\ \Lambda(\alpha^{-4}) &= \mathbf{1} + \alpha^6 + \alpha^2 = \mathbf{0} \\ \Lambda(\alpha^{-3}) &\neq \mathbf{0} \\ \Lambda(\alpha^{-2}) &\neq \mathbf{0} \\ \Lambda(\alpha^{-1}) &\neq \mathbf{0} \\ \Lambda(\alpha^{-0}) &\neq \mathbf{0}\end{aligned}$$

It follows that the erroneous components are \mathbf{r}_6 and \mathbf{r}_4 ; the set of the erroneous positions is $\mathbf{E} = \{6,4\}$.

The first order formal derivative of the $\Lambda(x)$ polynomial is:

$$\begin{aligned}\Lambda'(x) &= \sum_{i \in \mathbf{E}} \alpha^i \prod_{i \neq j} (\alpha^j x + \mathbf{1}) \\ \Lambda'(x) &= \alpha^6 (\alpha^4 x + \mathbf{1}) + \alpha^4 (\alpha^6 x + \mathbf{1}) = \alpha^6 + \alpha^4 = \alpha^3 \\ \Gamma(\alpha^{-6}) &= \Gamma(\alpha) = \alpha^6 \\ \Gamma(\alpha^{-4}) &= \alpha^2 \\ \mathbf{v}_6 &= \mathbf{r}_6 + \frac{\alpha^6 \alpha^6}{\alpha^3} = \mathbf{0} + \alpha^2 = \alpha^2 \\ \mathbf{v}_4 &= \mathbf{r}_4 + \frac{\alpha^4 \alpha^2}{\alpha^3} = \alpha + \alpha^3 = \mathbf{1}\end{aligned}$$

The corrected word is: $\mathbf{v} = (3 \ 5 \ 1 \ 4 \ 4 \ 3 \ 1)$.

RS coding and decoding using linear feedback shift registers LFSR

For binary codes there are various ways of implementing the encoder and decoder using LFSRs.

The RS codes are character oriented and require mathematical operations over $\text{GF}(2^k)$. The practical implementations require implementing these operations and expressing the characters in the Galois fields. All mathematical operations

over $GF(2^k)$ are done by specialised circuits that perform additions and multiplication over the same fields.

- *Addition circuit over $GF(2^k)$*

One possible circuit to realise the summation over $GF(2^k)$ is given in Fig. 5.26:

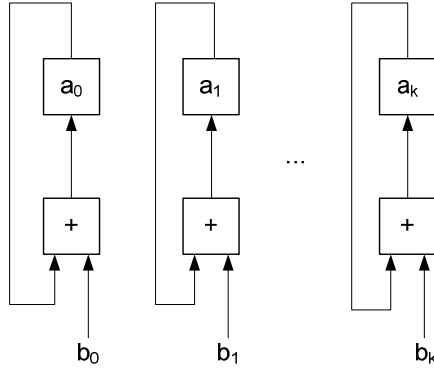


Fig. 5.26 Summing circuit over $GF(2^k)$

The addition of two elements from $GF(2^k)$ is done summing the binary numbers on k digit:

$$a = (a_0 \ a_1 \ \dots \ a_k)$$

$$b = (b_0 \ b_1 \ \dots \ b_k)$$

The circuit operates as follows:

1. Load the k cells of the register with the binary number on k digits corresponding to the element.
2. At each clock cycle the register is loaded with the adder parallel output.
3. Put the results in the register.

Example 5.23

Let us consider $\alpha^3, \alpha \in GF(2^3)$. The shift register will have 3 cells. The two elements are expressed as follows:

$$\alpha^3 = (0 \ 1 \ 1)$$

$$\alpha = (0 \ 1 \ 0)$$

Adding the two numbers, the register will contain at step 2 the following array:

$$\alpha^3 + \alpha = (0 \ 0 \ 1)$$

- *Multiplication circuit over GF(2^k)*

Multiplying two elements in Galois fields is, in fact, multiplying the polynomial representation of the two numbers (i.e. two polynomials of maximum degree k - 1) followed by a k-th order the modulo generator polynomial reduction.

Example 5.24

- *Multiplication circuit over GF(2⁴)*

Let us consider $b(x) = (b_0 \ b_1 \ b_2 \ b_3)$ and $a(x) = x$.

We have $b(x) = b_0 + b_1x + b_2x^2 + b_3x^3$ and $a(x) = x$

The generator polynomial of GF(2⁴) is: $p(x) = x^4 + x + 1$

$$\begin{aligned} a(x) \cdot b(x) &= b_0x + b_1x^2 + b_2x^3 + b_3x^4 = b_0x + b_1x^2 + b_2x^3 + b_3(x + 1) = \\ &= b_3 + (b_0 + b_3)x + b_1x^2 + b_2x^3 \end{aligned}$$

This relation determines the following way of implementation:

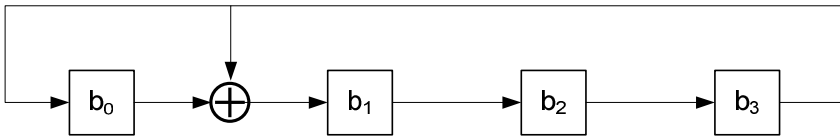


Fig. 5.27 Multiplying circuit with α over GF(2⁴)

There are several steps that need to be followed in order to get the final result:

1. Loading in parallel the register with the bits corresponding to the element b.
2. Shift to the right the register content.
3. The register parallel output gives the multiplication result.

In practice, high speed RS encoders/decoders are implemented using specialised circuits by Advanced Hardware Architecture Inc: the AHA 4510 series (e = 5), AHA 4010 (e = 10), AHA 4600 (e = 16).

Applications

Reed–Solomon codes are widely used for error protection in satellite transmissions, in storage on CDs and DVDs, applications in which powerful correcting codes are required in order to correct both independent and burst errors. In most applications RS codes are concatenated with convolutional codes.

Example 5.25

The block scheme of a CD player is given below:

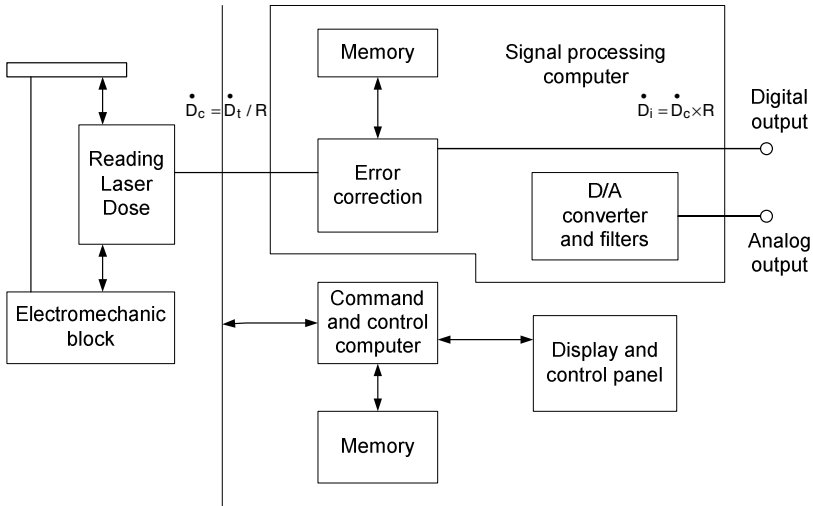


Fig. 5.28 CD player – block scheme

Among others, the command computer plays the songs following the options selected by the listener on the control panel.

When recording sounds on CDs, the error protection is realised using a RS(7,3) code.

Knowing that the sound bandwidth is 5Hz–20kHz and that sampling is done at Nyquist frequency, determine the information rate of the encoded source (under the assumption of equally probable source), as well as the capacity of the transmission channel at the digital output. Which is the capacity of a CD, if it can be played for 74 minutes? What is the decoder output sequence if the reading laser provides $\mathbf{r} = \begin{bmatrix} 0 & 2 & 0 & 4 & 2 & 0 & 0 \\ \text{MSC} \end{bmatrix}$?

Solution

- a. $f_s = 2f_{\max} = 40 \text{ kHz}$ – sampling frequency

$$C = \dot{D}_i \text{ (ideal assumption)}$$

$$\dot{D}_i = f_s m k = \dot{D}_c, \text{ where } m = \text{no. of information symbols and} \\ k \text{ the order of extension : } GF(2^3)$$

$$\dot{D}_i = f_s n_{\text{word}} = 0,84 \text{ Mb/s}$$

$$n_{\text{word}} = n \cdot k = 7 \cdot 3 = 21 \text{ bits}$$

$$C = 0,36 \text{ Mb/s}$$

- b. The codeword duration is given by the sampling period, so the transmission time for $n = 21$ bits is

$$T_s = \frac{1}{f_s} = 25\mu\text{s}$$

The memory capacity of a CD is determined by the decision quantity corresponding to the encoded information:

$$D_c = \frac{D_c}{t} \text{ so } D_c = 0,84 \text{ Mb/s} \cdot 74 \cdot 60 = 466,2 \text{ MB(megabytes)}$$

- c. Having used RS(7,3), one may easily determine that each character $v_i \in \text{GF}(2^3)$, so it will be expressed with $k = 3$ bits

We determine:

$$S_1 = \mathbf{r}(\mathbf{a}) = \mathbf{a}^3$$

$$S_2 = \mathbf{r}(\mathbf{a}^2) = \mathbf{a}^6$$

$$S_3 = \mathbf{r}(\mathbf{a}^3) = \mathbf{a}$$

$$S_4 = \mathbf{r}(\mathbf{a}^4) = \mathbf{a}^5$$

Due to the fact that $S_2^2 + S_1 S_3 = \mathbf{a}^{12} + \mathbf{a}^4 \neq \mathbf{0}$, the received sequence is affected by two errors ($t = 2$).

Using the formula from Table 5.13 to compute the σ_i coefficients, we get:

$$\sigma_1 = \mathbf{a}^3$$

$$\sigma_2 = \mathbf{a}$$

Chien search formula provides the erroneous positions \mathbf{r}_1 and \mathbf{r}_2 :

$$\mathbf{X}_1 = \mathbf{a}$$

$$\mathbf{X}_2 = \mathbf{a}^0 = \mathbf{1}$$

$$\mathbf{Y}_1 = \mathbf{1}$$

$$\mathbf{Y}_2 = \mathbf{a}^0 = \mathbf{1}$$

It follows that errors occur on control positions; the decoder output (Peterson algorithm with Chien search decoding) is:

$$\mathbf{i} = \begin{bmatrix} 0 & 2 & 0 \\ \text{MSC} \end{bmatrix} = [000|010|000]$$

5.9 Convolutional Codes

5.9.1 Representation and Properties

Convolutional codes have been introduced by P. Elias in 1955 as an alternative to block codes. Unlike the block codes, convolutional codes have encoding memory,

that is, at a certain moment of time, the n encoder outputs depend not only on the m inputs at that moment, but also on the previous M information blocks (an information block contains m bits). Important contributions to the development of these codes were brought by: J. Wozencraft who, in 1961, proposed the sequential decoding, J. Massay who presented in 1963 the threshold decoding and A. Viterbi who proposed in 1967 the minimum distance decoding algorithm, known as Viterbi algorithm.

These codes are used in practice due to some essential advantages: great detection and correction capacity for both independent and burst errors and, in many cases, simplicity in implementation. The disadvantage, due to the great redundancy, imply by the inefficient use of the band, which this is why these codes are used in applications for which the bandwidth is not critical, especially for satellite transmissions and space communications.

Figure 5.29 shows an intuitive presentation of these codes, in comparison with the block ones.

Convolutional codes – characteristic parameters

Constraint (M) represents the number of information blocks needed for determining a control symbol.

Constraint length (K) represents the number of information symbols needed for determining a control symbol.

$$K = M \cdot m \tag{5.212}$$

The *code distance* has the same significance as for block codes, with the remark that it is defined on a number of frames (blocks) N (used notation d_N instead of d).

Coding rate (R) has the same significance as for block codes (5.5), and it represents the relation between the number of information bits (m) and the length of an encoded block (n).









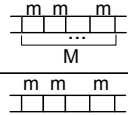
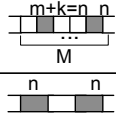
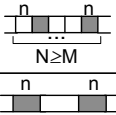
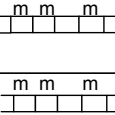
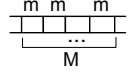
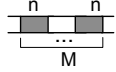
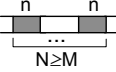
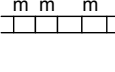
Code type		Encoder-input	Encoder-output	Decoder-input	Decoder-output
block	systematic				
	non-systematic				
convolutional	systematic				
	non-systematic				

Fig. 5.29 Comparison between block and convolutional codes

Remarks

The value of R is different for the two types of codes; for block codes R is big (>0.95) due to the low relative redundancy; for the convolutional codes R is usually small (typical values are $1/2$, $1/3$, $3/4$) reflecting their high relative redundancy.

As we have already shown in 5.1, the coded bit rate (\dot{D}_c) depends on R :

$$\dot{D}_c = \frac{\dot{D}_i}{R}$$

This justifies the need for a larger bandwidth, from where the possibility to use such encoding for transmissions on channels for which the bandwidth is not critical.

If the information sequence is of finite length, containing a number of L blocks, the encoded sequence is of length: $n(L+M)$. In this case the coding rate will be:

$$R = \frac{m \cdot L}{n(L+M)} \quad (5.213)$$

If $L \gg M$, then $L/(L+M) \cong 1$, so (5.213), valid for convolutional codes is identical with expression (5.5) for block codes.

If $R = 1/2$, it results $m = 1$, so basically, at the input the information is not divided in blocks but processed continuously, hence the name of *continuous codes*.

The name of *convolutional codes* comes from the fact that the k control symbols for systematic codes (the n symbols for non-systematic structures) are obtained from the digital convolution between the information sequence (i) and the generator polynomial (g).

5.9.2 Convolutional Codes Encoding

For convolutional codes each of the k control symbols (systematic codes) and each of the n symbols (for the non-systematic type) are obtained from K information symbols by multiplying the information sequence with the corresponding generator polynomials (Fig. 5.30).

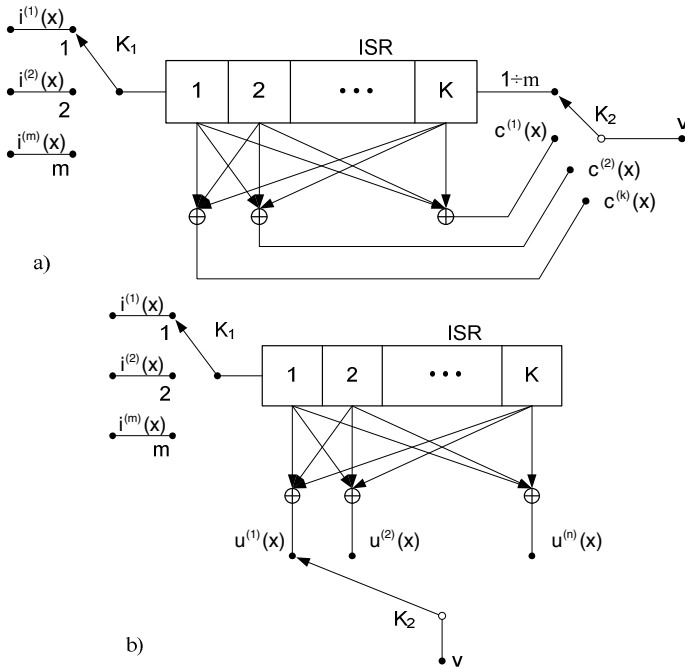


Fig. 5.30 Block scheme of: a) systematic and b) non systematic convolutional encoder; (ISR - Information Shift Register)

Unlike for block codes, for convolutional codes the information as well as the encoding process take place continuously; it follows that, for polynomial representation, the polynomials will be continuous.

$$\begin{aligned}
 i^{(1)}(x) &= i_0^{(1)} + i_1^{(1)}x + i_2^{(1)}x^2 + \dots \\
 i^{(2)}(x) &= i_0^{(2)} + i_1^{(2)}x + i_2^{(2)}x^2 + \dots \\
 &\vdots \\
 i^{(m)}(x) &= i_0^{(m)} + i_1^{(m)}x + i_2^{(m)}x^2 + \dots
 \end{aligned} \tag{5.214}$$

$$\begin{aligned}
 c^{(1)}(x) &= c_0^{(1)} + c_1^{(1)}x + c_2^{(1)}x^2 + \dots \\
 c^{(2)}(x) &= c_0^{(2)} + c_1^{(2)}x + c_2^{(2)}x^2 + \dots \\
 &\vdots \\
 c^{(k)}(x) &= c_0^{(k)} + c_1^{(k)}x + c_2^{(k)}x^2 + \dots \\
 u^{(1)}(x) &= u_0^{(1)} + u_1^{(1)}x + u_2^{(1)}x^2 + \dots \\
 u^{(2)}(x) &= u_0^{(2)} + u_1^{(2)}x + u_2^{(2)}x^2 + \dots \\
 &\vdots \\
 u^{(n)}(x) &= u_0^{(n)} + u_1^{(n)}x + u_2^{(n)}x^2 + \dots
 \end{aligned} \tag{5.215}$$

For this representation the bits succession at encoder input and output will be: (Fig. 5.31)

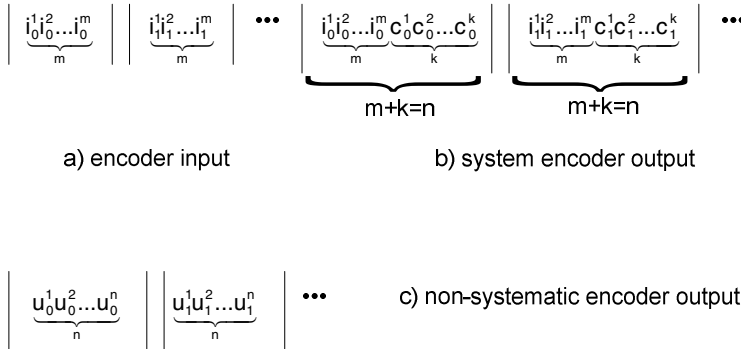


Fig. 5.31 Representation of the information at: a) encoder input; b) systematic encoder output; c) non-systematic encoder output

The control symbols $c^{(j)}(x)$ and the encoded symbols $u^{(i)}(x)$ are determined multiplying the information polynomial $i(x)$ with a generator polynomial $g^{(j)}(x)$, multiplication which represents the numerical convolution between \odot and $g^{(j)}$. The number of generator polynomials necessary for encoding is:

$$m \times k \tag{5.216}$$

for systematic codes and

$$m \times n \tag{5.217}$$

for non-systematic codes.

From these generator polynomials, at least one must be of degree $K-1$, due to the fact that we must use K information symbols to determine one $c^{(j)}$ or $u^{(j)}$ symbol:

$$g^{(j)}(x) = g_0^{(j)} + g_1^{(j)}x + \dots + g_k^{(j)}x^k + \dots, \quad k \leq K-1 \tag{5.218}$$

In this case, for a systematic code, we have:

$$c^{(j)}(x) = i(x)g^{(j)}(x) \tag{5.219}$$

or

$$\mathbf{c}^{(j)} = \mathbf{i} * \mathbf{g}^{(j)} \tag{5.219.a}$$

where $*$ represents the numerical convolution between \odot and g , all the operations being modulo 2. The term of order j of the convolution is:

$$c_1^{(j)} = \sum_{i=0}^k i_{1-i}g_i^{(j)} = i_1g_0^{(j)} + i_{1-1}g_1^{(j)} + \dots + i_{1-k}g_k^{(j)} \tag{5.220}$$

where $i_{1-i} = 0$ for $\forall 1 < i$.

In the same way, for a non-systematic code, we have:

$$\mathbf{u}^{(j)}(x) = \mathbf{i}(x) \mathbf{g}^{(j)}(x) \quad (5.221)$$

$$\mathbf{u}^{(j)} = \mathbf{i} * \mathbf{g}^{(j)} \quad (5.221.a)$$

Encoding relations (5.219) and (5.221) as well as their corresponding convolutions are linear, justifying why convolutional codes are linear codes.

One can easily built a convolutional encoder using a LSR with K cells and modulo two adders with inputs connected to the LSR according to the generator polynomials $\mathbf{g}^{(j)}(x)$.

Example 5.26

Be the convolutional code with parameters $R = 1/2$ and $K = 3$; encode the following information sequence in both cases, systematic and non-systematic: $\mathbf{i} = [0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0]$.
LSB

Solution

$R = m/n = 1/2$, so the codeword contains two symbols: one information symbol ($m = 1$) and one control symbol ($k = 1$) (for the systematic code).

Knowing that the constraint length is $K = m \times M = 3$ and $m = 1$, it results that in order to perform the encoding we need 3 information bits (blocks). The generator polynomials will have the maximum degree:

$$K-1 = 3-1 = 2$$

- systematic code: we have only one generator polynomial $m \times k = 1 \times 1 = 1$ and it must be of degree 2; we choose:

$$\mathbf{g}(x) = 1 + x^2 \text{ or } 1 + x + x^2$$

- non-systematic code: we have $m \times n = 1 \times 2 = 2$ generator polynomials, from which at least one must be of degree 2;

$$\mathbf{g}^{(1)}(x) = 1 + x^2$$

$$\mathbf{g}^{(2)}(x) = 1 + x + x^2 \text{ or } 1 + x$$

Remarks

- For non-systematic codes, depending on how we choose the generator polynomials, catastrophic errors may occur. An *error* is defined as being *catastrophic* if a finite number of errors in transmission produce an infinite decoding errors. One sufficient and necessary condition [42] to determine catastrophic errors (for codes with $R = 1/n$) is that generator polynomials have a common divisor; $\mathbf{g}^{(1)}(x) = 1 + x^2$ and $\mathbf{g}^{(2)}(x) = 1 + x$ have $1 + x$ as their common divisor, so this determines a *catastrophic code*.

$$1 + x^2 = (1 + x)(1 + x).$$

- Systematic codes can never be catastrophic and this is another advantage, besides their simplicity.

In what follows we will determine the encoded sequence for the given © in both encoding types using the encoding relations (5.219) and (5.221).

$$i(x) = x + x^2 + x^4$$

$$c(x) = i(x)g(x) = (x + x^2 + x^4)(1 + x^2) = x + x^2 + x^3 + x^6$$

The vector expression of the control sequence is:

$$\mathbf{c} = [0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 1]$$

For the systematic code, the encoded sequence is:

$$\mathbf{v} = \begin{bmatrix} 0 & 0 & | & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ i_0 & c_0 & & & & & & & & & & & & & & \end{bmatrix}$$

Using relation (5.222.a), we will determine the control sequence by numerical convolution:

$$\begin{aligned} \mathbf{c} &= \mathbf{i} * \mathbf{g} = \begin{pmatrix} i_0 & i_1 & i_2 & i_3 & i_4 & i_5 & i_6 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix} * \begin{pmatrix} g_0 & g_1 & g_2 \\ 1 & 0 & 1 \end{pmatrix} = \\ &= (0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 1) \end{aligned}$$

The components of the control sequence are determined with (5.220):

$$c_0 = i_0g_0 = 0$$

$$c_1 = i_1g_0 + i_0g_1 = 1$$

$$c_2 = i_2g_0 + i_1g_1 + i_0g_2 = 1$$

$$c_3 = i_3g_0 + i_2g_1 + i_1g_2 = 1$$

$$c_4 = i_4g_0 + i_3g_1 + i_2g_2 = 0$$

$$c_5 = i_5g_0 + i_4g_1 + i_3g_2 = 0$$

$$c_6 = i_6g_0 + i_5g_1 + i_4g_2 = 1$$

Similarly, we encode in the non-systematic case:

$$u^{(1)}(x) = i(x)g^{(1)}(x) = (x + x^2 + x^4)(1 + x^2) = x + x^2 + x^3 + x^6$$

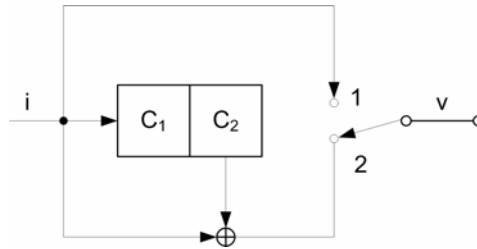
$$u^{(2)}(x) = i(x)g^{(2)}(x) = (x + x^2 + x^4)(1 + x + x^2) = x + x^5 + x^6$$

The code sequence will be:

$$\mathbf{v} = \begin{bmatrix} 0 & 0 & | & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ u_0^{(1)} & u_0^{(2)} & & & & & & & & & & & & & & & \end{bmatrix}$$

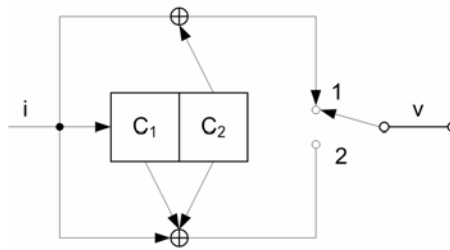
We suggest to the reader to determine the sequence \mathbf{v} using the convolution (see 5.221.a).

The encoders with LSRs and their operation is given in Fig. 5.32 a), and respectively 5.32 b).



T_n	t_{n+1}			T_n	
	i	C_1	C_2	V	
				$1(i)$	$2(i)$
1	0	0	0	0	0
2	1	1	0	1	1
3	1	1	1	1	1
4	0	0	1	0	1
5	1	1	0	1	0
6	0	0	1	0	0
7	0	0	0	0	1

a)



T	I	C_1	C_2	V	
				$1(u^{(1)})$	$2(u^{(2)})$
1	0	0	0	0	0
2	1	1	0	1	1
3	1	1	1	1	0
4	0	0	1	1	0
5	1	1	0	0	0
6	0	0	1	0	1
7	0	0	0	1	1

b)

Fig. 5.32 a) Convolutional systematic encoder - block scheme and operation; b) Convolutional non-systematic encoder - block scheme and operation

Remarks

- All the information sequences must be ended with $(K-1)$ zeros, necessary to decode the last frame corresponding to the given sequence as well as for flushing the encoding register (*trellis termination*).
- From the given example, one may notice the extreme simplicity of the convolutional encoders, a real advantage in their implementation.

In this example the information sequence length is small: $L = 5$ and for this reason the encoding rate R can not be calculated with (5.5), but with (5.213):

$$R = \frac{mL}{n(L+M)} = \frac{1 \cdot 5}{2(5+3)} = \frac{5}{16} = 0,31 < R = \frac{m}{n} = \frac{1}{2}$$

If we had a bigger length for the information sequence, for example $L = 500$, then the two relations would have been identical:

$$R = \frac{mL}{n(L+M)} = \frac{500}{2(500+3)} \cong \frac{1}{2} = \frac{m}{n}$$

A matrix description of the convolutional codes, using the control matrix \mathbf{H} or the generator matrix \mathbf{G} is given in [2], [28] and [43].

5.9.3 Graphic Representation of Convolutional Codes

A convolutional code can be graphically represented in many ways: with state diagrams, tree diagrams and trellis diagrams.

State diagram

A convolutional encoder is made up with a shift register having $K-1$ cells which define at one moment of time t_i , the encoder state $\mathbf{X}^{(i)} = (C_1^{(i)} \ C_2^{(i)} \ \dots \ C_{K-1}^{(i)})$ ($C_j^{(i)}$ denotes the state of the cell C_j at the moment i). Knowing the register state at moment i and the information symbol transmitted at $(i+1)$ we may determine the shift register state at the moment $(i+1)$. The code sequence generated at the moment i is completely determined the register state at the moment i : $\mathbf{X}^{(i)}$ and by the information symbol i_i , so the state $\mathbf{X}^{(i)}$ represents the encoder history:

$$\mathbf{X}^{(i)} = (i_{i-1} \ i_{i-2} \ \dots \ i_{i-K+1}) \quad (5.222)$$

The register evolution is a Markov chain, meaning that the transition from one state to another is determined only by the previous state:

$$P(\mathbf{X}^{(i+1)}/\mathbf{X}^{(i)}, \mathbf{X}^{(i-1)}, \dots, \mathbf{X}^{(0)}) = P(\mathbf{X}^{(i+1)}/\mathbf{X}^{(i)}) \quad (5.223)$$

The state diagram includes all possible states of the shift register as well as the encoded structure when transitioning from one state to another. There are only two

possible transitions from any state, corresponding to the emission of 0 or 1; it follows that it is not possible that during one transition to jump from a certain state in any other state.

Example 5.27

We will determine the state diagrams corresponding to the two codes given in Example 5.26.

The shift register used for encoding has $K-1 = 2$ cells, so it will have four distinct states: $a = 00$, $b = 10$, $c = 01$, $d = 11$

On each branch connecting two states are written the information symbol and the encoded structure: i/v .

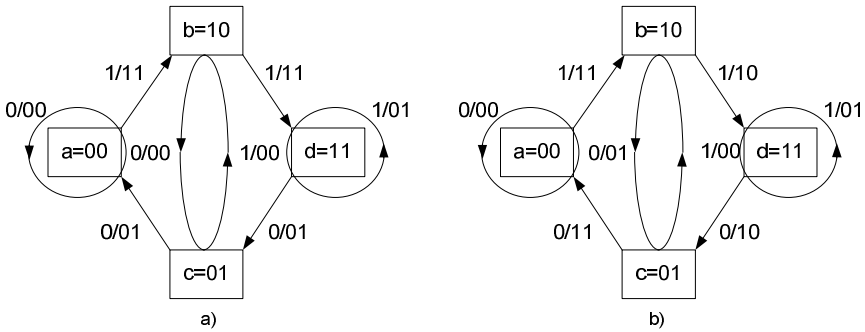


Fig. 5.33 State diagram of the convolutional code with $R = 1/2$, $K = 3$ for a) systematic and b) non-systematic type

Tree Diagram (Encoding Graph)

Although the state diagram completely describes the encoder, it does not give any information about its history. The encoding graph adds to the state diagram the time dimension.

The graph is drawn beginning with the initial state (t_0) zero of the shift register (state a). From this state two branches emerge, ongoing up and corresponding to the emission of a 0, and another one going down, corresponding to the emission of a 1. At the moment t_1 from each branch two other branches will emerge, corresponding to the emission of a 0 or 1 and so on. Adding the time dimension to the state diagram one may also know the decoder time evolution.

If the length of the information sequence (L) is big, the branches number grows exponentially: 2^L , and limits the use of the graph in practice.

Example 5.28

We will draw the graph corresponding to the systematic code from example 5.26 emphasizing the evolution of the encoder for the following information sequence:

$$\mathbf{i} = [0 \ 1 \ 1 \ 0 \ 1]$$

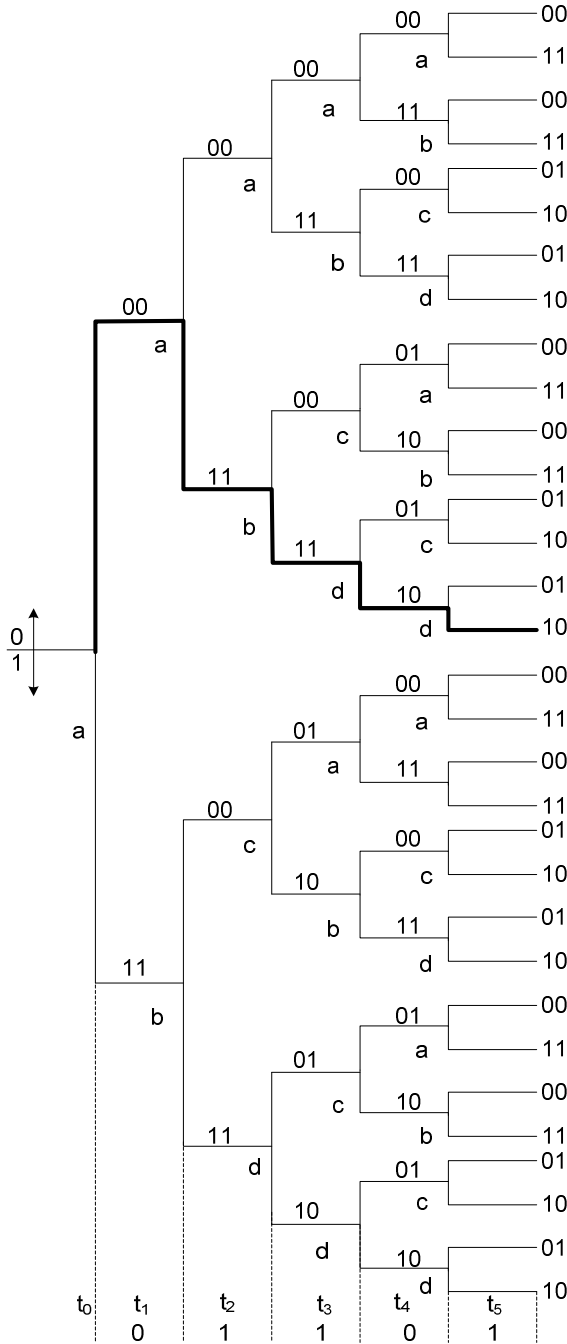


Fig. 5.34 The graph corresponding to the systematic convolutional code $R = 1/2$, $K = 3$; — the encoded structure for $\mathbf{i} = [0\ 1\ 1\ 0\ 1]$

Trellis Diagram

One may notice in Fig. 5.34 that the graph is repeating beginning with t_4 ; for the general case, the diagram repeats after K frames (K being the constraint length).

The first ramification (at moment t_1) leads to nodes a and b. At each next ramification the number of nodes doubles; four nodes occur at t_2 : a, b, c and d and eight at t_3 : 2 nodes a, 2 b, 2 c and 2 nodes d. It can be easily noticed that all the branches which emerge from the same state generate the same code sequences and this is the reason why the two diagram halves (superior and inferior) are identical. It follows that, in Example 5.28 ($K = 3$), the 4-th bit is fed to the encoder from the left, while the first bit (i_0) is removed from the register without having any influence on the codeword.

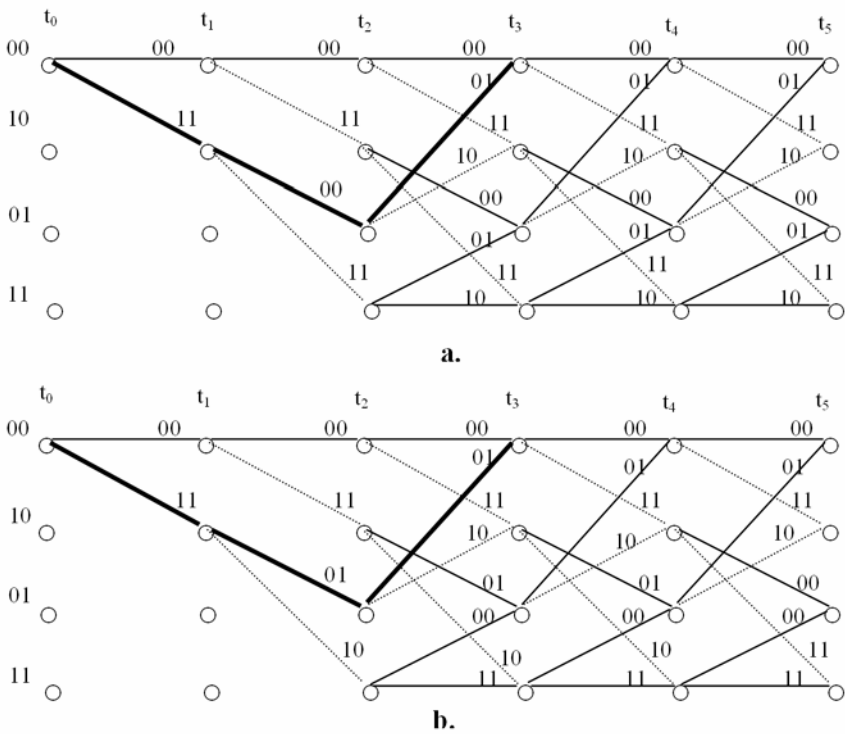


Fig. 5.35 Trellis corresponding to the convolutional code $R = 1/2$, $K = 3$: a) systematic with $g(x) = 1 + x^2$; b) systematic with $g(x) = 1 + x + x^2$; c) non-systematic with $g_1(x) = 1 + x^2$, $g_2(x) = 1 + x + x^2$

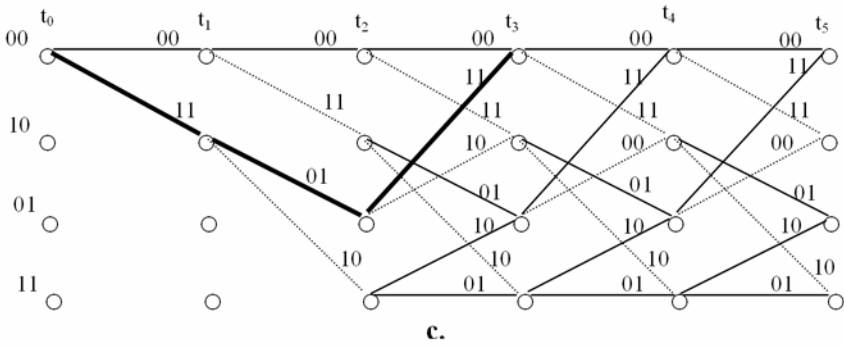


Fig. 5.35 (continued)

Subsequently, the sequences 0 1 1 x y and 1 1 1 x y generate the same codewords after $K = 3$ ramifications (frames). This means that any two nodes that have the same state at the same moment $t_i (i > K)$ can be connected as they generate identical sequences.

In this way, starting from the encoding graph we obtain another graphic representation called *trellis diagram*, from the trellis aspect of the diagram. In the trellis representation the encoded sequence will be marked with a continuous line when we apply “0”, and with a dot line if the input is “1”. The trellis nodes show the encoding register state. At a certain moment of time t_i the trellis contains 2^{K-1} nodes, determined by all register distinct states at encoding. Two branches enter each node starting with moment t_k (Fig. 5.35).

5.9.4 Code Distance and d_∞

Compared to block codes, for convolutional codes the code distance depends on the number of frames N used during the decoding process.

For a convolutional code the N -th order code distance (d_N) represents the minimum value of Hamming distance between any two possible code sequences on N frames and different in the initial frame.

$$d_N =: \min d_H(\mathbf{u}_N, \mathbf{v}_N) , \quad \mathbf{u}_1 \neq \mathbf{v}_1 \tag{5.224}$$

where \mathbf{u}_N and \mathbf{v}_N are two code sequences on N frames.

Remarks

- As for block codes, the branches containing exclusively zeros in the first frame are not taken into consideration
- Relation (5.16): $d = w_{\min}$ is valid
- d_N provides the same information about the code control capacity

The relation:

$$d_N \geq 2t+1 \tag{5.225}$$

shows that the code will correct any combination of t errors and less than t errors on N successive frames.

The minimum number of frames used in the decoding process is M (the constraint). In this case:

$$d_N = d_M = d_{\min} \tag{5.226}$$

represents the minimum code distance.

Unlike for block codes, the decoding process of a convolutional code can be realized using a number of frames N higher than the number of frames used during encoding (M); at limit $N \rightarrow \infty$ The code distance defined for convolutional codes in this last case is d_∞ or d_{free} and it is used for Viterbi or sequential decoding, where the decoding memory is basically, unlimited.

In order to define d_∞ [47], we search branches that start from the zero state and return to the same state (except for all zero branch in the first frame). The way with the minimum Hamming weight will be d_∞ . Obviously, $d_\infty \geq d_N$, which justifies the advantage of decoding on the number of frames on which d_∞ operates.

For most codes (with average or small K), d_∞ is obtained in several constraints or even in the first constraint. Practically, d_∞ (for high values of K) is obtained for $(4 \div 5)K$.

Example 5.29

We will determine $d_{\min} = d_K$ and d_∞ for the convolutional codes represented in Fig. 5.35.

- For the systematic code (a) we have: $R = 1/2$, $K = 3$, $g(x) = 1 + x^2$ and we determine:

Frame N	1	2	K = 3	Weight	d_K	d_∞
Sequence v	v_1	v_2	v_3	w_i		
	11	00	01	3	3	3
	11	00	10	3		
	11	11	01	5		
	11	11	10	5		

- For the systematic code (b) : $R = 1/2$, $K = 3$, $g_1(x) = 1 + x + x^2$

Frame N	1	2	K = 3	Weight	d_K	d_∞
Sequence v	v_1	v_2	v_3	w_i		
	11	01	01	4	3	4
	11	01	10	4		
	11	10	00	3		
	11	10	11	5		

- For the non-systematic code (c): $R = 1/2$, $K = 3$, $g_1(x) = 1 + x^2$ and $g_2(x) = 1 + x + x^2$

Frame N	1	2	K = 3	Weight	d_K	d_∞
Sequence v	v_1	v_2	v_3	w_i		
	11	01	11	5	3	5
	11	01	00	3		
	11	10	10	4		
	11	10	01	4		

One must notice that in all three cases, d_∞ is reached in the first constraint length $K = 3$.

For systematic codes, d_∞ is smaller than for non-systematic codes with identical parameters (R, K). Table 5.14 [42] presents a comparison between these codes from d_∞ point of view:

Table 5.14 d_∞ for the systematic and non-systematic codes: $R = 1/2$ and $K \in [2, 8]$

K	d_∞	
	systematic	non-systematic
2	3	3
3	4	5
4	4	6
5	5	7
6	6	8
7	6	10
8	7	10

5.9.5 Decoding (Viterbi Algorithm, Threshold Decoding)

Viterbi algorithm (1967)

This algorithm is based on the principle of minimum distance, which was shown in 5.6 that is obtained from maximum likelihood decoding (MLD), so it is an optimal algorithm.

Assume \mathbf{i} is a message encoded as \mathbf{v}_i (the code sequence) and received as \mathbf{r} . If all messages \mathbf{i} are equally probable, subsequently \mathbf{v}_i are equally probable, the error probability is minimum if the principle of maximum likelihood decoding is used.

If

$$p(\mathbf{r}/\mathbf{v}_i) > p(\mathbf{r}/\mathbf{v}_j), \quad i \neq j \quad (5.227)$$

then the receiver decides that \mathbf{r} comes from \mathbf{v}_i .

This decoding rule may be implemented quite easily. In 1967, Viterbi showed that for a BSC (a hard decoder), the decoding can be achieved by choosing for \mathbf{r} the sequence \mathbf{v}_i that has the minimum Hamming distance, thus using a minimum Hamming distance decoder. In 1969, Omura demonstrated that Viterbi algorithm is, in fact, the maximum likelihood decision (MLD) algorithm. In this case, relation (5.227) is equivalent to:

$$d_H(\mathbf{r}, \mathbf{v}_i) < d_H(\mathbf{r}, \mathbf{v}_j), \quad i \neq j \quad (5.228)$$

Viterbi algorithm operates on the trellis frame by frame, on a finite number of frames, in order to find the encoding path. At each node it calculates the distances between the received sequence and all the sequences on the trellis, cumulating the distance from one node to another (*cumulated distance*). In frame K , each node has two branches, so there will be two cumulated distances for each node; from these we keep the path with the minimum distance (*survivor*). If in a node there are two ways with equal distances, one of them is chosen randomly (the survivor). The frame $K+1$ is analyzed with the survivors from the node K , and so on. The analysis is continued on so many frames until there is only one path left; this path is considered the correct sequence.

The number of frames on which the decoding process takes place is called *decoding window* (W). W must be high enough in order to ensure the correct decision on the oldest transmitted frame.

Obtaining a unique route is a random variable. Computer simulations have shown that:

$$W \cong (4 \div 5)K \quad (5.229)$$

gives negligible inaccuracies compared to an infinite memory ($W \rightarrow \infty$) of the decoder; this is the reason for dimensioning the decoding windows of Viterbi decoders with relation (5.229).

Example 5.30

Applying Viterbi algorithm, decode the sequence \mathbf{r} given by:

$$\mathbf{r} = (11 \underline{11} 00 01 00 \underline{11} 00 01 11 00 00)$$

and knowing that the code parameters are: $R = 1/2$, $K = 3$, $g_1(x) = 1 + x^2$ and $g_2(x) = 1 + x + x^2$. Suppose \mathbf{r} contains two errors situated on the underlined positions.

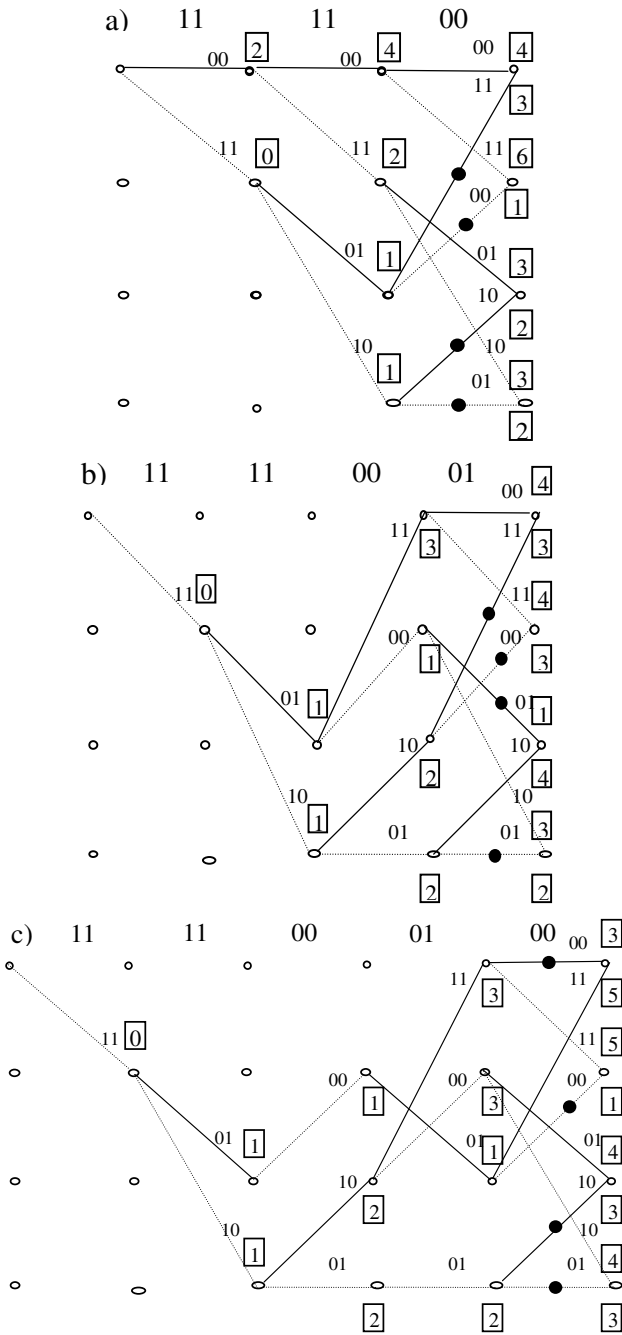


Fig. 5.36 Viterbi algorithm example for non-systematic convolutional code with $R=1/2$, $K=3$, $g_1(x)=1+x^2$, $g_2(x)=1+x+x^2$, on variable number of frames: $N=3 \div 12$.

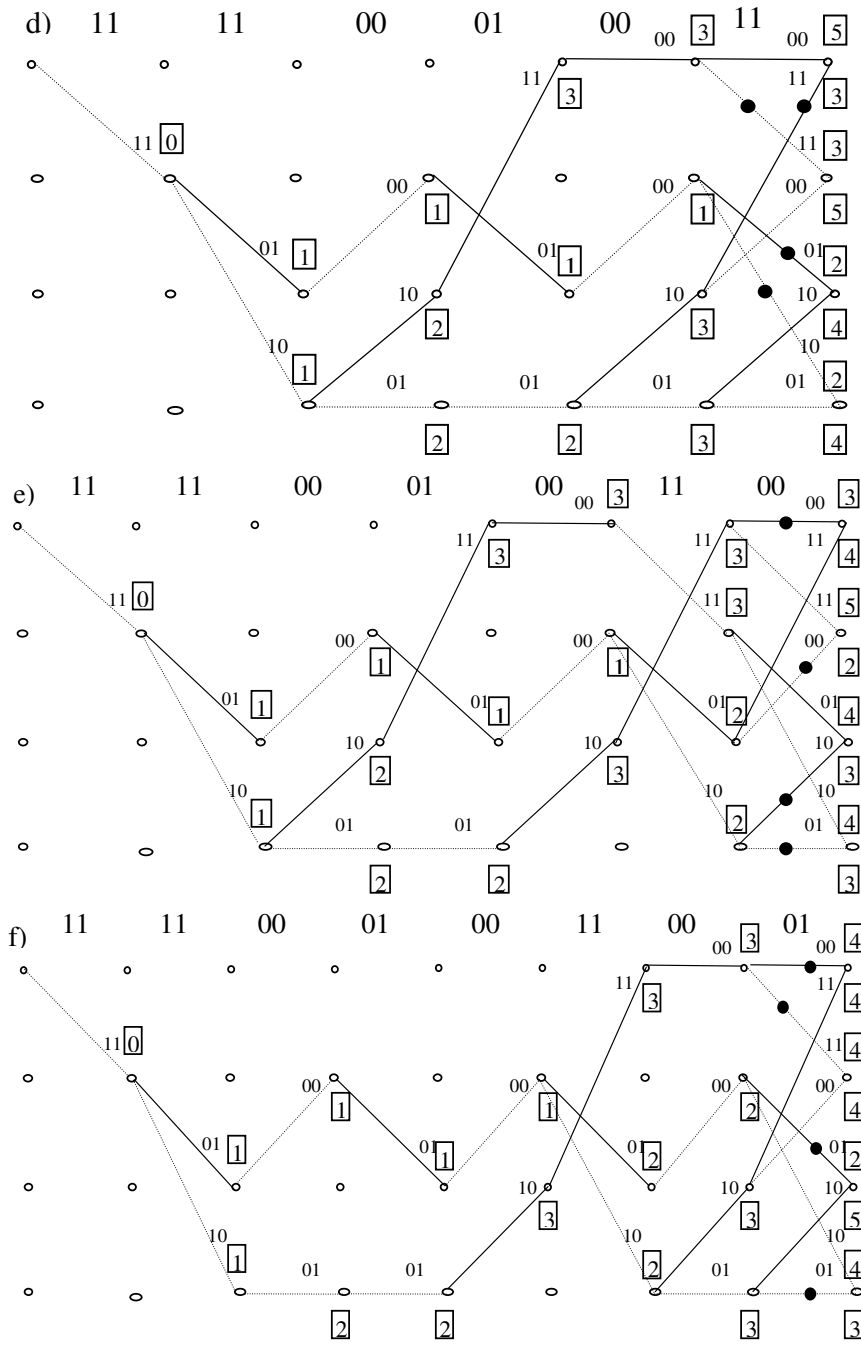


Fig. 5.36 (continued)

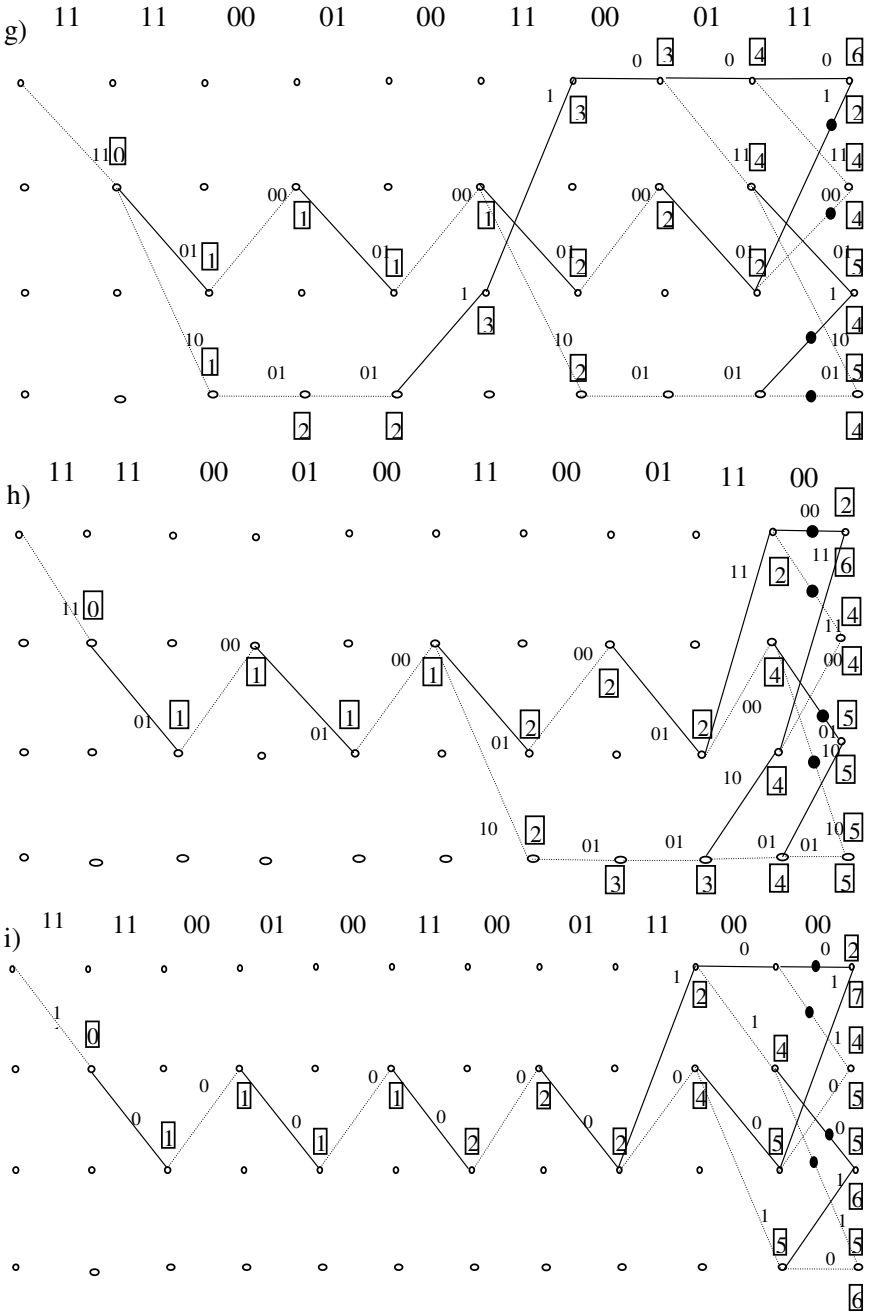


Fig. 5.36 (continued)

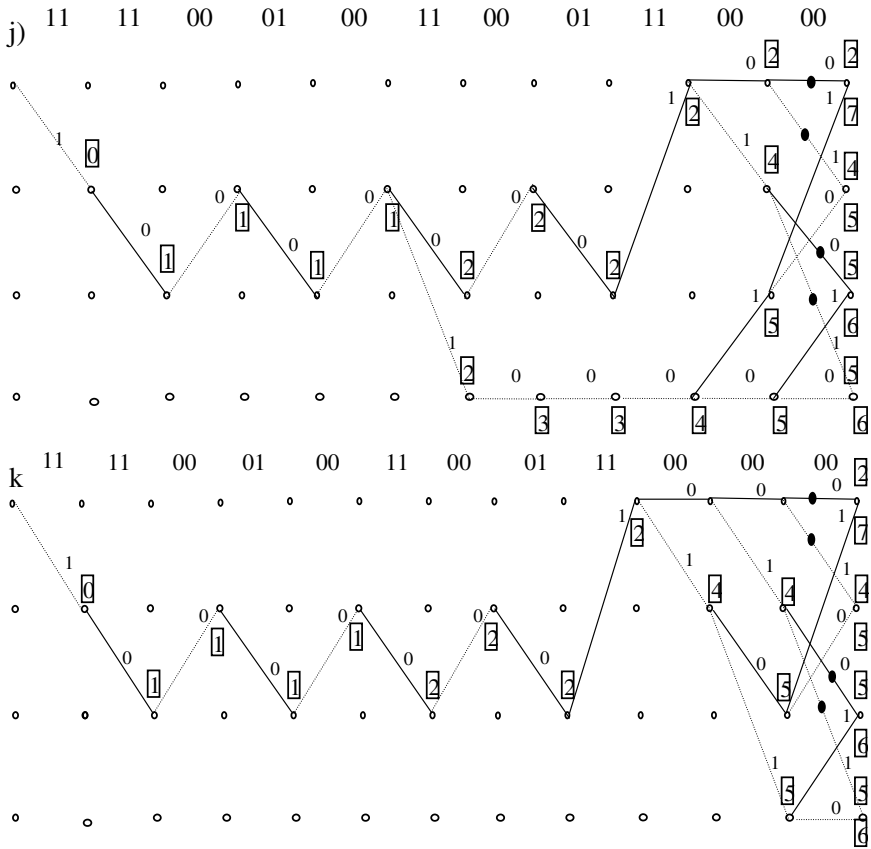


Fig. 5.36 (continued)

Solution

We will consider the trellis from Fig. 5.35.c. The decoding process will start by calculating the cumulated distances in each node between the received sequence and all the sequences on the trellis. After $K = 3$ frames, each node will contain two branches, from which only the survivors will be selected for the frame $K+1$ (corresponding to the minimum distances in each node). One must notice that in Fig. 5.36.c, there are two nodes that contain paths of equal distances, the survivor being chosen randomly. The same situation occurs in f, g, h, i; for Fig.5.36.h we illustrated two possible choices.

In Fig. 5.36.h, a unique route is obtained, after the first 5 frames, and 9 frames, respectively, in Fig.5.36.k. In this example, we analyzed the possibility of correcting two errors on twelve frames. The decoding window W has 12 frames, so $d_{\infty} = 5$ may be used; this implies that it is possible to correct any combination of two errors or less than two errors occurred on $W \cong (4 \div 5)K = 12 \div 15$ frames.

Example 5.31

In what follows assume the occurrence of three errors in the first constraint length.

$$\mathbf{r} = (\underline{10} \quad \underline{11} \quad \underline{01} \quad 00 \quad 11 \quad 01 \quad 00 \quad 01 \quad 00)$$

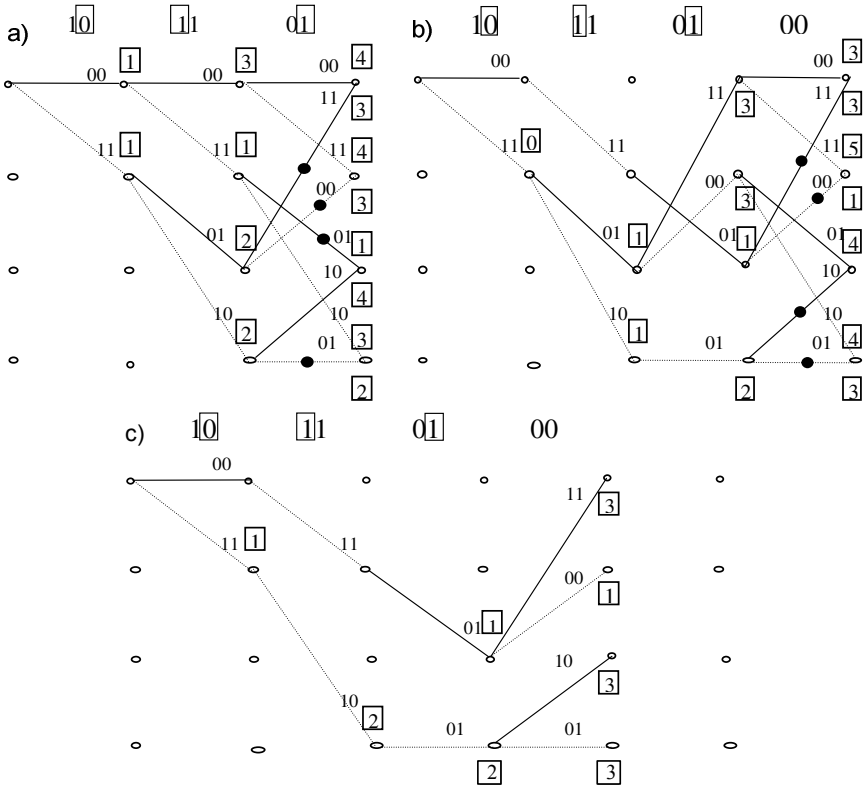


Fig. 5.37 Viterbi decoding process for a three error sequence in the first constraint length

From Fig. 5.37.c, we may notice that it is impossible to correct three errors on the first constraint length. For convolutional codes, the correction capacity cannot be expressed as easily as for block codes, due to the fact that it depends on the error distribution.

Soft Decision Decoding

The algorithm that has just been described corresponds to the transmission on a BSC, so on a memoryless channel. This model (BSC) corresponds to a *hard decision channel*, which means that, although the signals received by the demodulator are continuous (Gaussian variables, due to the Gaussian noise), at the demodulator output the signals are binary (the decision is binary or hard [28], [43]). If the

demodulator output is quantified by more than two levels, the decoding process is called *soft (decision) decoding* process (Fig. 5.38).

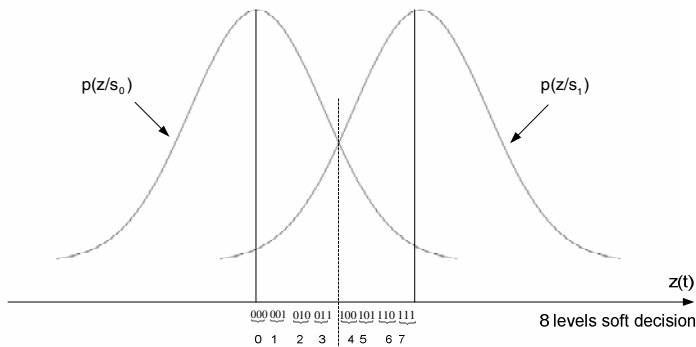


Fig. 5.38 Graphical illustration of the hard decision (with 2 levels) and the soft decision (with 8 levels)

For hard decoding process, the demodulator sends to the decoder only one symbol (0 or 1). For soft decoding with 8 quantizing levels, the demodulator sends to the decoder 3 bits, for each T time sample, which is equivalent with transmitting to the decoder a measure of the trust/confidence degree. Thus, transmitting the code 111 is equivalent with the transmission of a “1” with a very high degree of confidence, while transmitting 100 signifies the fact that “1” has a very low confidence. For the soft decoding process, Hamming distances are replaced with Euclidian distances between the received word and all possible code words [47]. Examples are given in 5.11 (turbo codes).

For a Gaussian channel and 8 levels (3 bits) soft decoding, the encoding gain is increased by 2 dB, compared to hard decoding; the coding gain for analog decision (with an infinite number of quantizing levels) is 2,2 dB, which means that for soft decision with 8 levels there is a loss of only 0,2 dB. This argument justifies, in the soft decision case, the use of maximum 8 levels. The price paid for 2 dB gain is the increased dimension of the memory needed in the decoding process (and possible speed losses as well) [42]. Soft decoding is frequently used in the Viterbi algorithm because it produces a slight increase in the computations volume [47].

Conclusions regarding Viterbi decoding algorithm

- it is an optimal algorithm
- it can be applied to any convolutional code (systematic or non-systematic)
- it can be used with hard decision or 8 quantizing levels (the encoding gain obtained in this case is of 2 dB compared to the hard decoding)
- there are VLSI specialized circuits: for example the Q 1401 circuit manufactured by Qualcomm [38] is a Viterbi decoder for a convolutional code with $R = 1/2$ and $K = 7$, operating with hard or soft decoding with 8

quantification levels, ensuring a $\dot{D}_i = 17\text{Mb/s}$, $G_c = 5,2 \text{ dB}$ at $p = 10^{-5}$ with BPSK or QPSK modulator and 8 levels soft decoding.

- the practical implementation is limited by K up to maximum values of approx. 10 (the calculation volume increases exponentially with K).

Threshold decoding (1963)

Threshold decoding, although not optimal compared to the Viterbi decoding, is an algebraic method of decoding, conceptually closer to block codes (the error syndrome is calculated in a similar manner), having the advantage of a very simple implementation. At the beginning it was used for BCH decoding and it was proposed by Massay in 1963 for convolutional decoding.

The decoding process takes place at the end of the constraint length K , which leads to inferior performances than those obtained with Viterbi algorithm (we remind that $d_K < d_\infty$, so the code control capacity on length K is reduced too).

The implementation great simplicity imposed this method in a series of applications that accept lower coding gains, but also low cost: in telephony and satellite radio transmissions [28].

The algorithm can be applied only to systematic convolutional codes.

The block-scheme of a threshold decoding system is presented in Fig. 5.39.

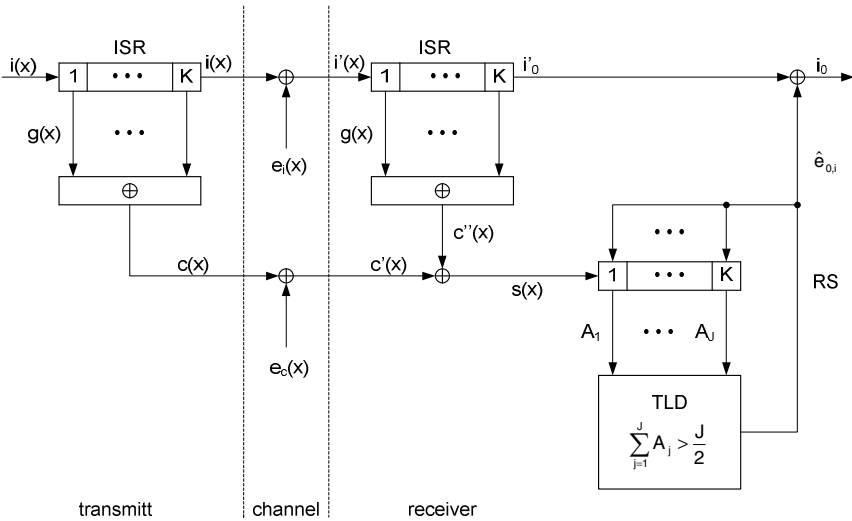


Fig. 5.39 Block-scheme of a threshold decoding system

The transmission was represented in parallel for information $i(x)$ and control $c(x)$ sequences in order to enable an easier understanding of the decoding process (in real serial transmission, the information and control bits are affected by errors independently).

In Fig. 5.39 are illustrated the followings:

- ISR - information shift register with K cells in which the information symbols are loaded; these symbols are then used for determining the control symbols
- $e_i(x)$ - error polynomial that alters the information symbols in the encoded sequence
- $e_c(x)$ - error polynomial that changes the control symbols in the encoded sequence
- $c''(x)$ - polynomial corresponding to the control symbols determined from the received information symbols $i'(x)$
- $s(x)$ - error syndrome polynomial
- $i'(x)$ - polynomial corresponding to the received information
- $c'(x)$ - polynomial corresponding to the received control bits
- TLD- threshold logic device (majority logic device); its output is "1" if the most of the inputs A_1, \dots, A_J are "1". If:

$$\sum_{j=1}^J A_j > \frac{J}{2} \quad (5.230)$$

then the output is "1".

SR - syndrome register; it is a K cells shift register

Correction starts from the first symbol introduced in ISR (i_0), the TLD output being "1" if i_0 is erroneous.

The equations describing the operation of the scheme are the followings:

$$i'(x) = i(x) + e_i(x) \quad (5.231)$$

$$c'(x) = c(x) + e_c(x) \quad (5.232)$$

$$c''(x) = g(x)i'(x) = g(x)[i(x) + e_i(x)] \quad (5.233)$$

$$c(x) = g(x)i(x) \quad (5.234)$$

$$\begin{aligned} s(x) &= c'(x) + c''(x) = c(x) + e_c(x) + c(x) + g(x)e_i(x) \\ s(x) &= e_i(x)g(x) + e_c(x) \end{aligned} \quad (5.235)$$

From (5.235), we notice that the error syndrome does not depend on the transmitted word, but only on the error word: $e_i(x)$ and $e_c(x)$ and on $g(x)$.

The convolutional codes, as we have already seen, are continuous codes so their polynomial representation will be made with infinite polynomials:

$$s(x) = \sum_{n=0}^{\infty} s_n x^n \quad (5.236)$$

Depending on $g(x)$, the convolutional codes are divided into direct orthogonal codes (DOC) and *indirect orthogonal codes* (IOC).

We call *direct orthogonal codes* (DOC) the codes that allow to directly determine a set of J equations s_j that are orthogonal on a given symbol.

A set of J equations is orthogonal on a given symbol if all the equations contain that symbol and any other symbol cannot be found in more than one equation, at the most.

Example 5.32

Analyze the threshold decoding algorithm for the convolutional code given by the following parameters: $R = 1/2$, $K = 7$, $g(x) = 1 + x^2 + x^5 + x^6$

Solution

We express the error syndrome in polynomial form, according to relation (5.235):

$$s(x) = (1 + x^2 + x^5 + x^6)e_i(x) + e_c(x) \quad (5.237)$$

The n -th order coefficient of the polynomial $s(x)$ will be:

$$s_n = e_{n,i} + e_{n-2,i} + e_{n-5,i} + e_{n-6,i} + e_{n,c} \quad (5.238)$$

The first 7 coefficients: s_0, \dots, s_6 will be loaded in SR (syndrome register).

We determine the following system:

$$\begin{cases} s_0 = e_{0,i} + e_{0,c} \\ s_1 = e_{1,i} + e_{1,c} \\ s_2 = e_{2,i} + e_{0,i} + e_{2,c} \\ s_3 = e_{3,i} + e_{1,i} + e_{3,c} \\ s_4 = e_{4,i} + e_{2,i} + e_{4,c} \\ s_5 = e_{5,i} + e_{3,i} + e_{0,i} + e_{5,c} \\ s_6 = e_{6,i} + e_{4,i} + e_{1,i} + e_{0,i} + e_{6,c} \end{cases} \quad (5.238.a)$$

From the 7 equations of this system we identify equations s_0, s_2, s_5, s_6 being orthogonal on i_0 ; all of them contain $e_{0,i}$ and any other symbol is not found in more than one equation:

$$\begin{cases} A_1 = s_0 = e_{0,i} + e_{0,c} \\ A_2 = s_2 = e_{0,i} + e_{2,i} + e_{2,c} \\ A_3 = s_5 = e_{0,i} + e_{3,i} + e_{5,i} + e_{5,c} \\ A_4 = s_6 = e_{0,i} + e_{1,i} + e_{4,i} + e_{6,i} + e_{6,c} \end{cases} \quad (5.239)$$

The terms A_1, \dots, A_4 are the $J = 4$ inputs of the threshold logic device TLD. We wish to evaluate the correction capability of this code.

- Assume one error on position $i'_0 : e_{0,i} = 1$. We get $A_1 = A_2 = A_3 = A_4 = 1$, so the TLD output, according to (5.230), is “1”; it follows that i'_0 will be corrected.
- Assume two errors: one on i'_0 and another position, either information or control; in this case, one of the equations $A_j = 0$, the other three are “1”, so i'_0 will be corrected.
- If three errors occur: i'_0 and another two on either information or control symbols, only one or two A_j inputs will be “1”; the output of the majority logic device will be “0”, therefore i'_0 will not be corrected.

Resuming, we may say that the maximum number of errors that can be corrected (t_{max}) is:

$$t_{max} = \frac{J}{2} \tag{5.240}$$

Remark

$g(x)$ must be carefully chosen, such that it does not decrease the code correction capacity given by (5.241):

$$d_K \geq 2t + 1 \tag{5.241}$$

It follows that:

$$J = d_K - 1 \tag{5.242}$$

The block scheme of the threshold decoder is the following:

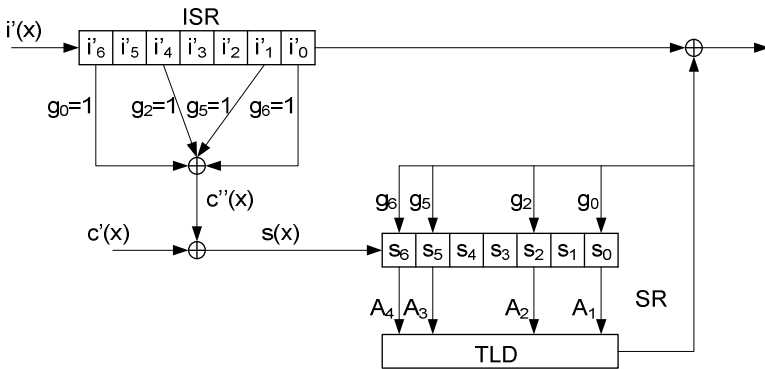


Fig. 5.40 Threshold decoder for the direct orthogonal code $R = 1/2$, $g_2(x) = 1 + x^2 + x^5 + x^6$

The generator polynomials for direct orthogonal codes DOC can be easily determined using the control triangles [2], [47].

Table 5.19 contains the most important DOCs for $R = 1/2$:

Indirect orthogonal codes (IOC) are the codes for which the orthogonal system of equations on a given symbol cannot be obtained directly, but by linear combinations of the syndrome equations. These codes are more efficient than the DOCs in the way that the same J (so the same power of correction) is ensured by a lower K (these polynomials are determined by computer trials - 1963 Massey). It must be underlined the fact that the syndrome register SR must be used with feedback.

Table 5.15 The most important direct orthogonal codes [47] for $R = 1/2$

J	K	$g(x)$
2	2	$1+x$
4	7	$1+x^2+x^5+x^6$
6	18	$1+x^2+x^7+x^{13}+x^{16}+x^{17}$
8	36	$1+x^7+x^{10}+x^{16}+x^{18}+x^{30}+x^{31}+x^{35}$

Example 5.33

Analyze the threshold decoding for the indirect orthogonal code given by $R = 1/2$, $K = 6$, $g(x) = 1 + x^3 + x^4 + x^5$.

Solution

The n -th order coefficient of the polynomial $s(x)$ is:

$$s_n = e_{n,i} + e_{n-3,i} + e_{n-4,i} + e_{n-5,i} + e_{n,c} \tag{5.243}$$

The first six coefficients loaded in the syndrome register are:

$$\left\{ \begin{array}{l} s_0 = \underline{e_{0,i}} + e_{0,c} \\ s_1 = e_{1,i} + e_{1,c} \\ s_2 = e_{2,i} + e_{2,c} \\ s_3 = e_{3,i} + e_{0,i} + e_{3,c} \\ s_4 = e_{4,i} + e_{1,i} + e_{0,i} + e_{4,c} \\ s_5 = e_{5,i} + e_{2,i} + e_{1,i} + \underline{e_{0,i}} + e_{5,c} \end{array} \right. \tag{5.244}$$

The system (5.244) is not an orthogonal system on i_0 , although the equations s_0, s_3, s_4 and s_5 all contain $e_{0,i}$ and s_4 and s_5 contain $e_{1,i}$. In order to make it orthogonal, we combine equations s_1 and s_5 . Thus we obtain an orthogonal system on i_0 :

$$\begin{cases} A_1 = s_0 = \underline{e_{0,i}} + e_{0,c} \\ A_2 = s_3 = e_{3,i} + \underline{e_{0,i}} + e_{3,c} \\ A_3 = s_4 = e_{4,i} + e_{1,i} + \underline{e_{0,i}} + e_{4,c} \\ A_4 = s_1 + s_5 = \underline{e_{0,i}} + e_{2,i} + e_{5,i} + e_{1,c} + e_{5,c} \end{cases} \quad (5.245)$$

The block-scheme of the decoding unit is that illustrated in Fig.5.41 and the most important indirect orthogonal codes for $R = 1/2$ are given in Table 5.16 [47].

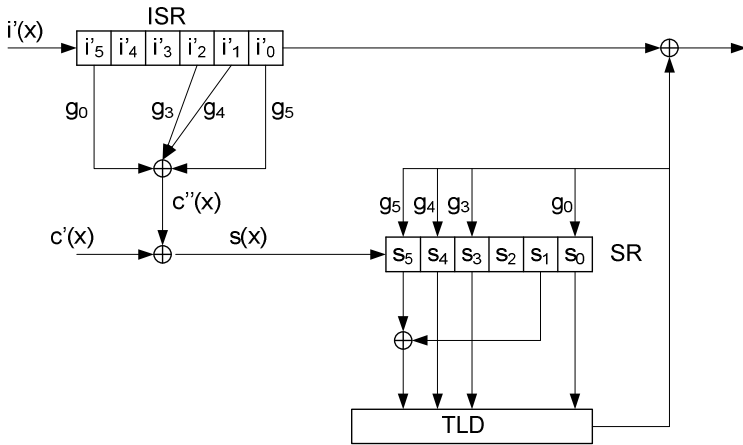


Fig. 5.41 Threshold decoder for indirect orthogonal code $R = 1/2$, $g_2(x) = 1 + x^3 + x^4 + x^5$

Table 5.16 The most important indirect orthogonal codes for $R = 1/2$

J	K	A_i	$g(x)$
4	6	$s_0, s_3, s_4, s_1 + s_5$	$1 + x^3 + x^4 + x^5$
6	12	s_0, s_6, s_7, s_9 $s_1 + s_3 + s_{10}$ $s_4 + s_8 + s_{11}$	$1 + x^6 + x^7 + x^9 + x^{10} + x^{11}$
8	22	$s_0, s_{11}, s_{13}, s_{16}, s_{17}$ $s_2 + s_3 + s_6 + s_{19}$ $s_4 + s_{14} + s_{20}$ $s_1 + s_5 + s_8 + s_{15} + s_{21}$	$1 + x^{11} + x^{13} + x^{16} + x^{17} + x^{19} + x^{20} + x^{21}$

Remark

Comparing Table 5.15 and Table 5.16, we remark the advantage of the indirect orthogonal codes: at the same correction capacity J, their K values are much lower.

The convolutional codes can also be decoded sequentially. This method is in fact the first proposed method for convolutional codes decoding and was proposed in 1957 by J. M. Wozencraft. Basically, a sequential decoder works by generating hypotheses regarding the transmitted code sequences; it measures the distances between these hypotheses and the received signal and goes on until the distances remain reasonable. When these are too high, there is a turn back, so a modification of the hypotheses, until, through attempts, the suitable hypotheses are found [28], [42]. From the sequential decoding process advantages, we mention the independence from the constraint length, which enables the use of this method for high K values ($K = 41$) [42]. The main disadvantage is that the number of the “weak” hypotheses and the turn backs depend on the channel signal/noise ratio (SNR). For low SNRs, the number of hypotheses that have to be tested is higher than for high values of this ratio, which in certain cases lead to exceeding the decoding storage capacity.

5.10 Code Interleaving and Concatenation

5.10.1 Interleaving

The process of *interleaving* allows the use of independent error correction codes for burst error correction of lengths much higher than the number of independent errors which occur in a received word.

The *idea* is to send the codeword symbols interleaved with other codewords symbols, such that the distance between two successive symbols of the same codeword is higher than the length of the burst (b). In this way, a burst of errors cannot affect more symbols in the same word and subsequently, a certain codeword symbols are erroneous due to different bursts (independent bursts); their effect is that of the independent errors (Fig. 5.42).

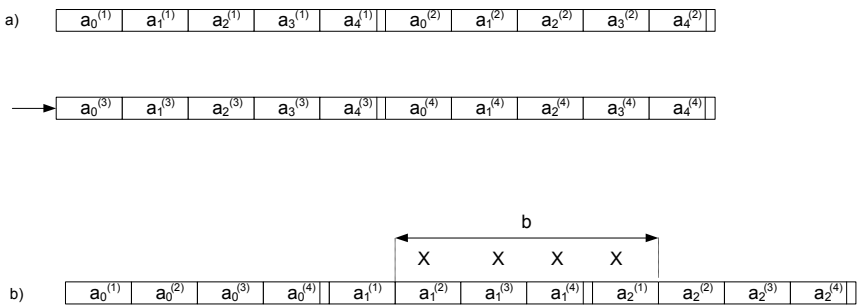


Fig. 5.42 Interleaving example: a) 4 codewords (of length 5) non-interleaved succession; b) interleaved succession ($b =$ burst error length)

Interleaving can be made in two ways: block interleaving and convolutional interleaving [42].

Block interleaving

When transmitting, the symbols of an independent error correcting block code are disposed column by column (or line by line) in a block of N lines and M columns and then sent to the modulator on lines (or columns). At the receiver the operations occur in reverse order: the received symbols enter the demodulator by lines and are sent to the decoder by columns (or viceversa).

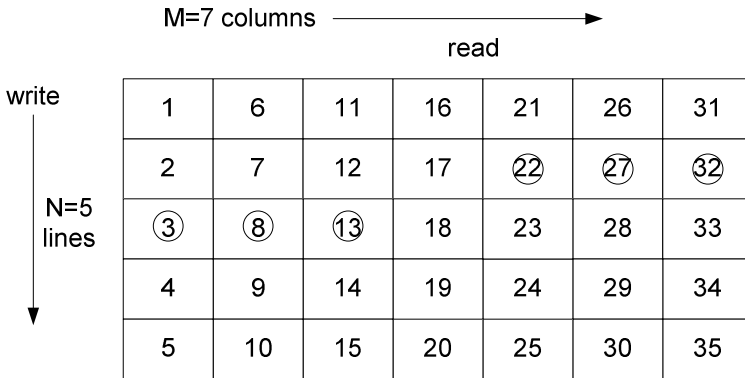


Fig. 5.43 Block interleaving

Block interleaving characteristics are:

- Any burst error of length $b \leq M$ will lead to singular errors at the decoder input.
- The delay caused by interleaving (taking into account the processing that takes place both at transmission and receiver) is approximately $2 \cdot N \cdot M \cdot T$ (T is the symbol duration). More precisely, in order to initiate the transmission (immediately after loading the first symbol of the last column: 31 in Fig. 5.43), it is enough to load the memory in $N(M-1)+1$ cells. The same minimum number of cells is necessary to initiate the decoding at the reception. It results that the entire minimum delay (at the transmission and reception) is: $2NM - 2N + 2$
- The necessary memory must have $N \cdot M$ cells; generally, the memory implemented both at the transmission and receiver is double: $2 \cdot N \cdot M$, allowing to simultaneously load one block and flush another.
- If the block code corrects only singular errors, the number of block columns M is chosen higher than the error burst length: $M \geq b$
- The number of lines (N) depends on the type of code used: for block codes: $N > n$ (the block code length), and for convolutional codes $N > K$ (the constraint length). In this way, a burst error of length $b = M$ will determine one error, at the most, in one code word, or in any constraint length K . If we use t errors correcting codes, M is chosen such that: $M > b/t$

The *convolutional interleaving* was proposed by Ramsey (1970) [59] and Forney (1971) [60]. The structure proposed by Forney is presented in Fig. 5.44.

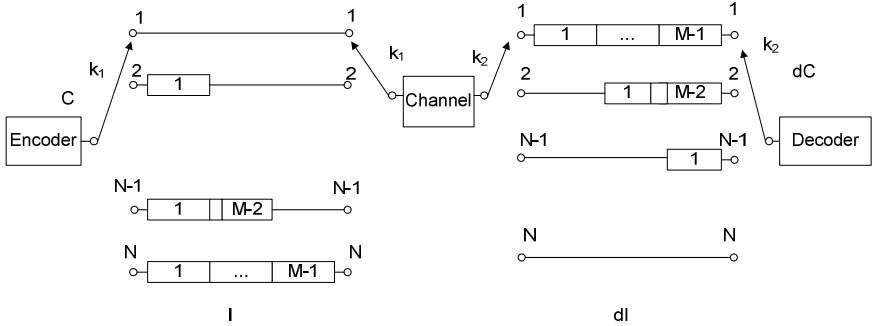


Fig. 5.44 Block-scheme of a convolutional interleaving made with shift registers (C-encoder for independent errors; dC-decoder for independent errors; I – interleaver; dI – de-interleaver).

The characteristics of convolutional interleaving are similar to those of block interleaving. The essential advantage of convolutional interleaving is that the total delay (transmission-receiver) is half of the block interleaving delay: $N(M-1)/2$, so the memory is reduced to half as well.

5.10.2 Concatenated Codes

Concatenated codes are using two levels: an inner C_1 and an outer C_2 (Fig. 5.45).

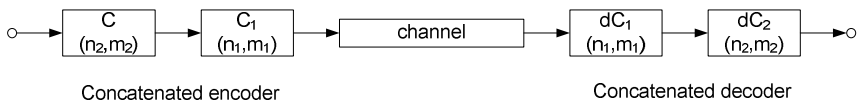


Fig. 5.45 Block scheme of a concatenated system

The resulting code is of dimension (n_1n_2, m_1m_2) , and its code distance is $d \geq d_1d_2$, where d_1 and d_2 are the corresponding code distances for C_1 and C_2 [28].

Generally, the outer code C_2 is a RS (Reed-Solomon) code, and C_1 is binary (usually a convolutional code) or RS as well.

Concatenated codes are useful when dealing with mixed errors: independent and burst errors. The inner code C_1 is usually dimensioned such that it corrects most of the independent errors on the channel. The outer code C_2 reduces the error probability to the desired value (it corrects some of the errors missed by C_1).

The purpose of concatenating is to obtain a low bit error rate with simplicity in implementation (lower than the complexity required by the use of a single code), and higher redundancy, i.e. a more powerful code.

One of the most popular concatenated systems is C_2 RS code, C_1 - convolutional code (implemented with Viterbi soft decoding) [28] (Fig. 5.46), for which $p = 10^{-5}$ corresponding to $E_b/N_0 = (2 \div 2,5)\text{dB}$ (the planetary NASA standard for codes).

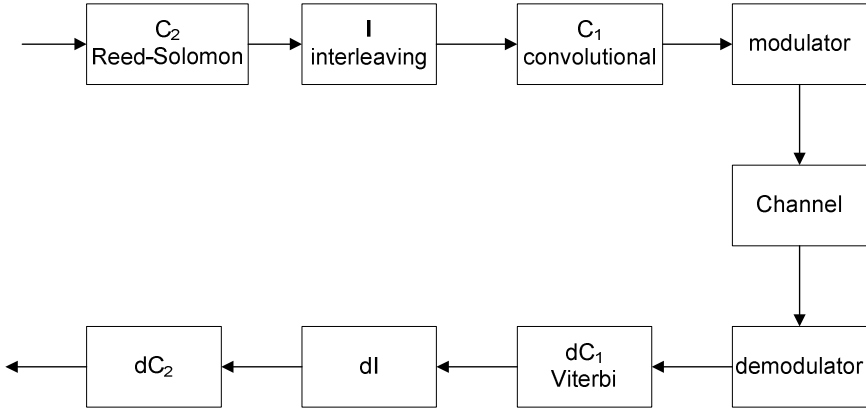


Fig. 5.46 Block-scheme of an interleaved concatenated system (C_2 - RS + C_1 - convolutional)

Example 5.34

The digital audio encoding system for CD (CIRC) [42]

In 1979, Philips Corporation from Netherlands and Sony from Japan defined a standard for the audio digital system using CDs.

The CD is a plastic disc of 120 mm used for audio signals storage. The sampling is performed at $f_s = 44.1\text{ kHz}$ (corresponding to an audio bandwidth of 20 kHz). Each audio sample is quantized with 2^{16} levels (16 bits or 2B bytes per sample); it results a dynamic of 96dB/s and harmonic distortions of 0,005% . One disc ($\cong 70$ minutes) contains 10^{10} bits. The disc is laser written and read.

The error sources on a CD are:

- small, unwanted air bubbles inside the plastic or irregularities in recording;
- fingerprints or scratches when handling the disc

These errors are similar to burst errors, as they affect more bits. The high fidelity of the system is due to a correction scheme based on concatenating two interleaved RS codes (CIRC - Cross Interleave Reed-Solomon Codes). By interleaving, data are redistributed such that the digits belonging to neighbouring samples are spread in space; subsequently, the burst errors will occur as independent (singular) errors. The error protection is ensured by shortened RS codes.

In digital audio applications, an undetected error is very important because it generates cracking noises, whereas detected errors are not so annoying as they can be “hidden”.

The block-scheme of the CIRC system is presented in Fig. 5.47

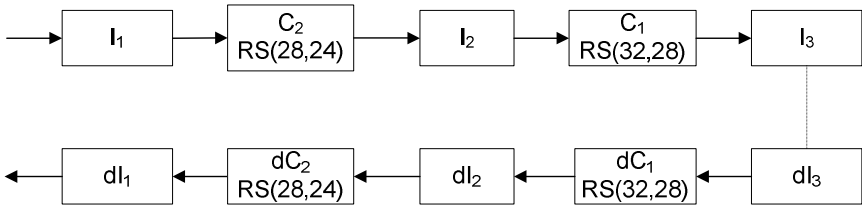


Fig. 5.47 Block scheme of CIRC system

The CIRC system controls the errors in a hierarchical structure:

1. The decoder provides a certain level of correctable errors;
2. If the error correction capacity is exceeded, the decoder provides a correction level of erasures;
3. If the level of erasures is exceeded, the decoder tries to mask the erroneous samples by interpolating the neighbouring unaltered samples;
4. If the interpolation capacity is exceeded, the decoder turns on the mute option during the deteriorated samples.

Fig. 5.48 and Fig. 5.49 illustrate the processes that occur at encoding and decoding, respectively.

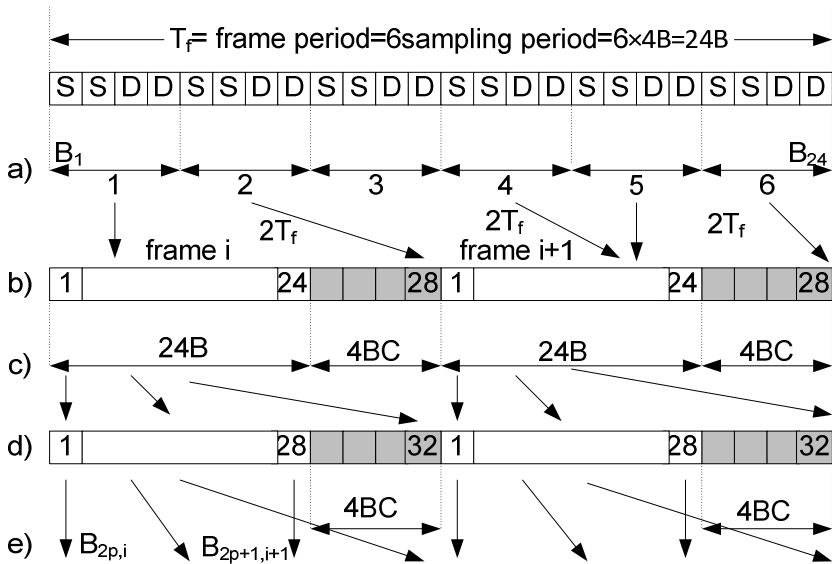


Fig. 5.48 Encoding and interleaving in CIRC system: a) I_1 - even samples B_{2p} are separated from the odd ones B_{2p+1} with $2T_f$; b) C_2 - RS (28,24) encoding; c) I_2 - samples are delayed with different time periods to spread the errors; d) C_1 -RS (32, 28) encoding; e) I_3 - even samples ($B_{2p,i}$) cross interleaving with the next frame odd samples ($B_{2p+1,i+1}$)

Fig. 5.48 (a) shows that a frame includes six sampling sequences, each sample made up of a stereo pair (left L, right R) quantized on 16 bits (2B). This results in a frame length of 24B. At encoding, the processing has five steps, as follows:

- a. I_1 - even samples are separated from the odd ones with two frame lengths ($2T_f$) to “mask” the detectable but not correctable errors without difficulty, by interpolation (due to the correct neighbouring samples).
- b. C_2 - it is an RS (28,24) encoder: 4 control bytes (4BC) are added to 24B. The RS (28,24) code is a shortened code: $n = 2^k - 1 = 2^8 - 1 = 255$: RS(255,251). We remind that by shortening, the code maintains its control capacity ($d = 5$), which corresponds to the correction of maximum 2 erroneous characters (bytes).
- c. I_2 - each of the 28B is differently delayed, so that the errors in one word are spread into different words (interleaving I_2). C_2 and I_2 have the function of correcting burst and independent errors that cannot be corrected by C_1 .
- d. C_1 - is an RS(32,28) encoder, derived just like C_2 by shortening the RS(255,251) code; therefore the codes have the same distance $d = 5$.
- e. I_3 - performs the cross interleaving between the even bytes ($B_{2p,i}$) of one frame and the odd bytes of the next frame ($B_{2p+1,i+1}$). Using this method, two consecutive bytes belong to different codewords.

C_1 and I_3 will correct most of the singular errors and will detect the long burst errors.

The processing that takes place at the decoder (player) is shown in Fig. 5.49.

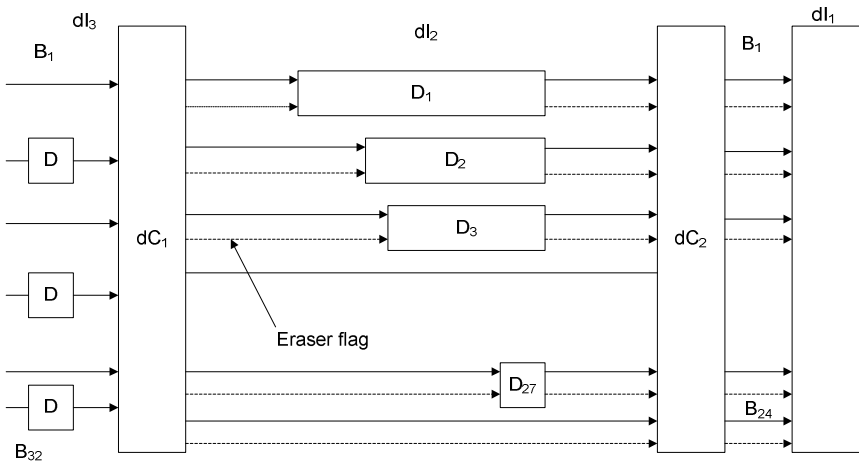


Fig. 5.49 Illustration of the decoding process in CIRC system

At receiver, the processing has also five steps, as follows:

- a. dI_3 – de-interleaving is accomplished by the alternate introduction of delay cells D . The 32 inputs ($B_1 \div B_{32}$) are loaded in parallel at the input of the de-interleaving circuit. The delays D take one byte.
- b. dC_1 - RS(28) decoder; dI_3 and dC_1 are designed for correcting one character in a block of 32B and detecting long burst errors. If multiple errors occur, the dC_1 decoder leaves them uncorrected, attaching to the 28B an erasure flag transmitted with dotted lines (see Fig. 5.49).
- c. dI_2 - the delay lines D_1, \dots, D_{27} spread the errors on a number of words at the dC_2 decoder input, which reduces the number of errors on a C_2 codeword, so that C_2 can correct them.
- d. dC_2 - corrects the burst errors missed by dC_1 . If the errors cannot be corrected but only detected by dC_2 , the errors pass unchanged through dI_1 ; however, dI_1 associates them an erasure flag (dotted lines $B_1, \dots B_{24}$).
- e. dI_1 - the uncorrected but detected errors (flagged errors) will be masked by interpolation from the correct neighbouring samples.

Interpolation and mute interpolation

The samples uncorrected by dC_2 cause distortions that can be heard as cracks. The interpolator dI_1 inserts new samples from the unaltered neighbouring samples, replacing the incorrect ones. If burst errors with $b > 48$ frames occur and two or more successive samples are not corrected, the system goes mute for several ms, undetectable by the human ear [32], [42].

5.11 Turbo Codes

5.11.1 Definition and Encoding

Turbo-codes (TC), invented in 1993 by Claude Berrou and Alain Glavieux [5] at ENST (Telecomme) Bretagne, France, are a novel and revolutionary error-control coding technique, almost closing the gap between Shannon limit and real code performance.

Turbo code consists of a parallel concatenation of two recursive systematic convolutional codes.

A *recursive convolutional code* is a convolutional code (with feed forward) provided with feedback too.

A *recursive systematic convolutional (RSC) code* is a systematic recursive convolutional code of rate $R=1/n$. A RSC with $R=1/2$ is given in Fig. 5.50.

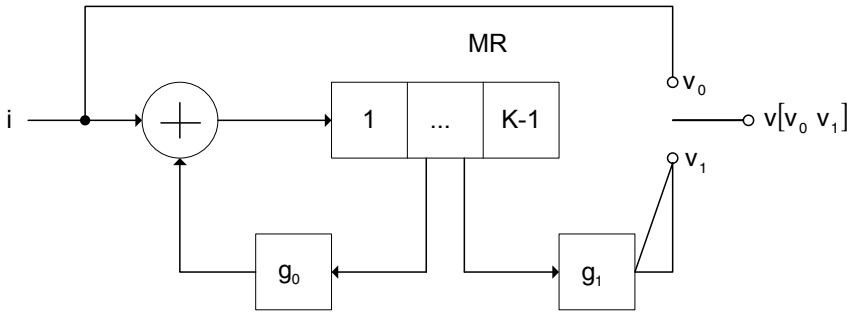


Fig. 5.50 Basic RSC code: $R=1/2$, g_0 - feedback polynomial, g_1 - feed forward polynomial

In a polynomial representation, we have:

$$v_0(x) = i(x) \tag{5.246}$$

$$v_1(x) = i(x) \frac{g_1(x)}{g_0(x)} \tag{5.247}$$

or:

$$\begin{aligned} v(x) &= [v_0(x), v_1(x)] = \left[i(x), i(x) \frac{g_1(x)}{g_0(x)} \right] = \\ &= i(x) \left[1, \frac{g_1(x)}{g_0(x)} \right] = i(x) \mathbf{G}(x) \end{aligned} \tag{5.248}$$

where $\mathbf{G}(x)$, the generator matrix, is identified as :

$$\mathbf{G}(x) = \left[1, \frac{g_1(x)}{g_0(x)} \right] \tag{5.249}$$

The variable x is representing the delay operator (D).

If the coding rate is $R = \frac{1}{n}$, the generator matrix of the corresponding $\frac{1}{n}$ RSC code is:

$$\mathbf{G}(x) = \left[1, \frac{g_1(x)}{g_0(x)}, \dots, \frac{g_{n-1}(x)}{g_0(x)} \right] \tag{5.250}$$

If the polynomials $g_i(x)$ ($i \geq 1$) and $g_0(x)$ are relatively prime (no common factor), the ratio $\frac{g_i(x)}{g_0(x)}$, $g_0(x)$ can be written as [48]:

$$\frac{g_i(x)}{g_0(x)} = a_0 + a_1x + a_2x^2 + \dots \tag{5.251}$$

and corresponds to the encoded sequence $v_i(x)$, ($i \neq 0$), the control sequence.

For iterative decoding, $g_0(x)$ - the feedback polynomial is selected to be a *primitive* polynomial of degree $K-1$ (the memory of the encoder).

In this parallel concatenation a random interleaver is used exploiting Forney concatenation idea. The information bits for the second code are not transmitted, thus incrementing the code rate (R).

The main reason to use a long interleaver is to generate a concatenated code with a large block length which leads to a large coding gain [46].

A basic turbo coder of $R=1/3$ with BPSK (Binary Phase Shift Keying) modulation and AWGN noise is presented in Fig 5.51.

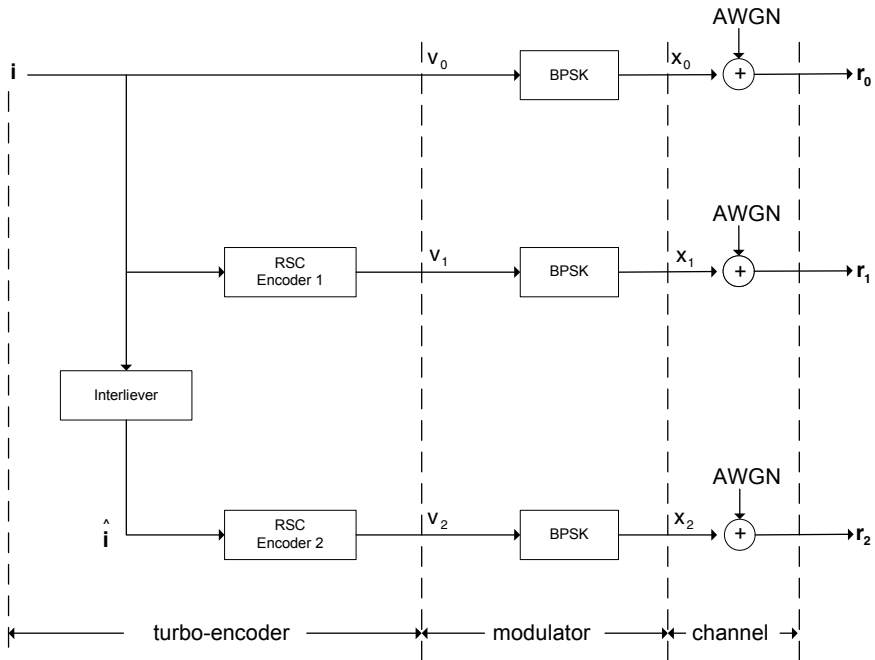


Fig. 5.51 Basic turbo transmitter: turbo encoder with $R=1/3$, BPSK modulation and AWGN channel.

5.11.2 Decoding

5.11.2.1 Basic Principles

A turbo decoder (Fig. 5.52) consists of two serially concatenated decoders separated by the same interleaver. The decoding algorithm is an iterative one (MAP or SOVA- soft output Viterbi algorithm). The iterative process performs information exchange between the two decoders. Increasing the number of iterations in turbo-decoding, a BER as low ($10^{-5} \div 10^{-7}$) can be achieved at a SNR very close to Shannon limit (-1.6 dB).

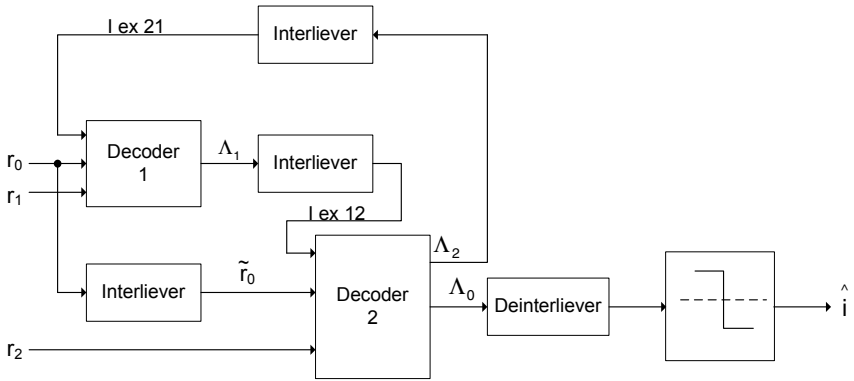


Fig. 5.52 Basic turbo-decoder

The first decoder has as inputs r_0 and r_1 . The decoder 1 produces a soft output Λ_1 which is interleaved and used to produce an improved estimate of the a priori probabilities of the information sequence (I_{ex12} *extrinsic information* of the first decoder is the priori probability estimate of the second decoder). The decoder performance can be improved by this iterative operation relative to a single operation of the serial concatenated decoder. After a number of iterations, the soft outputs of both decoders stop to produce further performance improvements and the last decoding stage makes a hard decision after deinterleaving.

The feedback loop is a distinguishing feature to the decoder and the name “turbo” is given precisely to the similarities, as concept and performance, with the thermodynamic turbo engine [19]. The two decoders cooperate similar with the pistons of a turbo engine: each decoder supplies its pair with extrinsic information, as the pistons supply each other with mechanical energy through the turbine.

Iterative decoding algorithms are numerous and complex: Viterbi, SOVA, MAP, LOG-MAP and Max-LOG-MAP [46], [21]. Briefly they will be presented in what follows.

5.11.2.2 Viterbi Algorithm (VA) [46], [48]

This algorithm, presented in 5.9.5, was originally proposed for convolutional codes decoding.

Decoding time is assumed τ , and the sequence starts from zero state (at moment 0) and reach the same state at the final moment (this is why the trellis termination is required).

$$S = (S_0 = 0, S_1, \dots, S_{\tau-1}, S_\tau = 0)$$

VA estimate the information \hat{i} that corresponds to the modulated sequence $\hat{\mathbf{x}}$ in the trellis, such that the word (sequence) error probability: P_s is minimized.

$$P_s = 1 - P_c \tag{5.252}$$

where P_c is the probability of a correct sequence decoding:

$$P_c = \int_{\mathbf{r}} p(\mathbf{r})p(\mathbf{i}/\mathbf{r})d\mathbf{r} \quad (5.253)$$

\mathbf{r} being the space of received sequences.

P_s minimum, implies P_c maximum, and according to (5.253) $p(\mathbf{i}/\mathbf{r})$, the a posteriori probability maximum. The algorithm which Maximizes the Aposteriori Probability is known as *MAP decoding algorithm*.

Using Bayes formula (see relation (2.32)) we have:

$$p(\mathbf{i}/\mathbf{r}) = \frac{p(\mathbf{i})p(\mathbf{r}/\mathbf{i})}{p(\mathbf{r})} = p(\mathbf{r}/\mathbf{i}) \quad (5.254)$$

based on the fact that the a priori probability of the information sequence $p(\mathbf{i})$ is equal with $p(\mathbf{r})$ (encoding is a biunivoque correspondence: $p(\mathbf{i}) = p(\mathbf{r})$).

A decoder which maximizes $p(\mathbf{r}/\mathbf{i})$ is called *maximum likelihood decoder* (MLD) (see also 5.6). Under the assumption of equally probable \mathbf{r} (equally probable sequences), MAP and MLD are equivalent in terms of word (sequence) error probability.

Under the assumption of AWGN with σ_n^2 dispersion (see Appendix D), and τ length sequence, we have:

$$p(\mathbf{i}/\mathbf{r}) = p(\mathbf{r}/\mathbf{i}) = \prod_{t=1}^{\tau} p(\mathbf{r}_t/\mathbf{i}_t) = \prod_{t=1}^{\tau} \prod_{k=0}^{n-1} \frac{1}{\sqrt{2\pi\sigma_n^2}} e^{-\frac{(r_{t,k} - x_{t,k})^2}{2\sigma_n^2}} \quad (5.255)$$

Taking into account that in relation (5.255) there are exponentials, the $\log p(\mathbf{i}/\mathbf{r})$ will be used:

$$\begin{aligned} \log p(\mathbf{i}/\mathbf{r}) &= \sum_{t=1}^{\tau} \log \prod_{k=0}^{n-1} \frac{1}{\sqrt{2\pi\sigma_n^2}} e^{-\frac{(r_{t,k} - x_{t,k})^2}{2\sigma_n^2}} = \\ &= -\frac{n\tau}{2} \log 2\pi - n\tau \log \sigma_n - \sum_{t=1}^{\tau} \sum_{k=0}^{n-1} \frac{(r_{t,k} - x_{t,k})^2}{2\sigma_n^2} \end{aligned} \quad (5.256)$$

Maximizing $p(\mathbf{r}/\mathbf{i})$ is equivalent, based on (5.256) to minimizing the Euclidian distance:

$$d_E = \sqrt{\sum_{k=0}^{n-1} (r_{t,k} - x_{t,k})^2} \quad (5.257)$$

It follows that, for AWGN channels, the MLD (maximum likelihood decoder) and obviously MAP and VA, reduce to a minimum distance (Euclidian) decoder (the same demonstration, but for hard decoding, using Hamming distance, is given in 5.6, relation (5.24)).

The squared Euclidian distance is called *branch metric* (mb_t^x) and is:

$$mb_t^x = \sum_{k=0}^{n-1} (r_{t,k} - x_{t,k})^2 \quad (5.258)$$

The *path metric* (mp_t^x) corresponding to the sequence x is:

$$mp_t^x = \sum_{j=1}^t mb_j^x = mp_{t-1}^x + mb_t^x \quad (5.259)$$

Thus, VA is an effective way of finding a path in the trellis with the minimum path metric. The computation is based on keeping only one path per node, the one with the minimum metric at each time instant (the *survivor*).

The decision on the message estimation \hat{i} is made at the final time τ . The maximum likelihood path is chosen as the survivor in the final node. If the minimum path metric corresponds to a path \hat{x} , the decoder will select the binary sequence \hat{i} on this path as the hard estimate of the transmitted sequence i .

Example 5.35

- Design a RSC encoder with coding rate $R=1/2$ and constraint length $K=2$.
- Encode the information sequence: $i=[0 \ 1 \ 0 \ 0 \ 1]$
- Draw the state diagram and the trellis
- If the received sequence is: $r=[1, -1|0.8, 1| -1, 1| -1, 1| 1, -1]$, applying VA with Euclidian distance, find the maximum likelihood path.

Solution

- From the input data $\frac{1}{2}$ RSC with $K=2$, we identify:

$$R = \frac{m}{n} \Rightarrow m=1, n=2 \Rightarrow k=1$$

According to (5.216) it follows that only one generator polynomial is required and K being 2, it means that its degree is 1:

$$g_0(x) = 1 + x \quad \text{and} \quad G(x) = \left[1, \frac{1}{1+x} \right]$$

The block scheme of the RSC encoder is given in Fig. 5.53a.

b) The encoding can be done algebraically, using (5.246), (5.247) or (5.248); the same result is obtain using the encoder operation (Fig. 5.53).

– Algebraic encoding:

$$i(x) = x + x^4 = v_0(x)$$

$$v_1(x) = i(x) \frac{1}{g_0(x)} = \frac{x^4 + x}{x + 1} = x + x^2 + x^3$$

$$v = [v_0 v_1] = [00|11|01|01|10]$$

The encoder operation (Fig. 5.53) is:

t_n	t_{n+1}	t_n	
		$v = [v_0 v_1]$	
i	C_1	$v_0 = i_n$	$v_1 = i_n \oplus C_{1,n-1}$
0	0	0	0
1	1	1	1
0	1	0	1
0	1	0	1
1	0	1	0

Remark: the encoder starts and ends at zero state (S_0)

c) The state diagram and the trellis are represented in Fig. 5.53.b, respectively c.

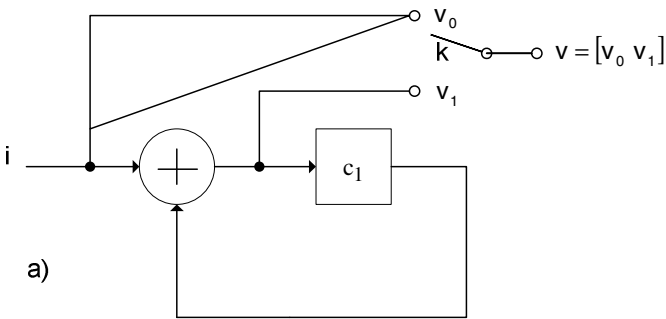


Fig. 5.53 RSC encoder with $R = 1/2$ and $K=2$: a) block- scheme; b) state-diagram; c) trellis: — $i=0$ input, $i=1$ input

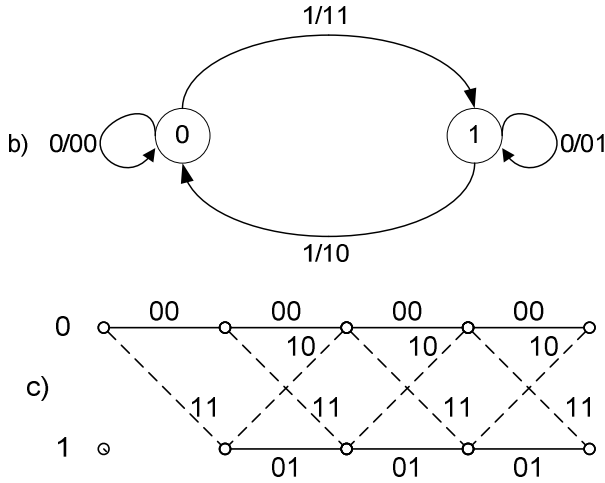
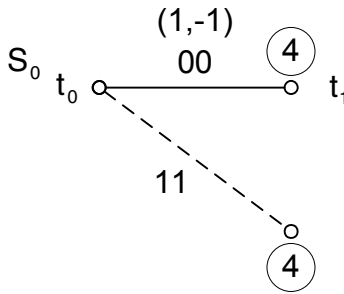


Fig. 5.53 (continued)

- d) We calculate the branch metric mb_t^x in each frame and starting with the second frame, the path metric mp_t^x . Taking into account that from the second frame in each node enter two metrics, the survivor will be selected based on its minimum Euclidian distance:
- In the first frame, the branch metrics are:



$$mb_1^1 = [1 - (-1)]^2 + [-1 - (-1)]^2 = 4 = mp_1^1$$

$$mb_1^2 = [1 - 1]^2 + [-1 - 1]^2 = 4 = mp_1^2$$

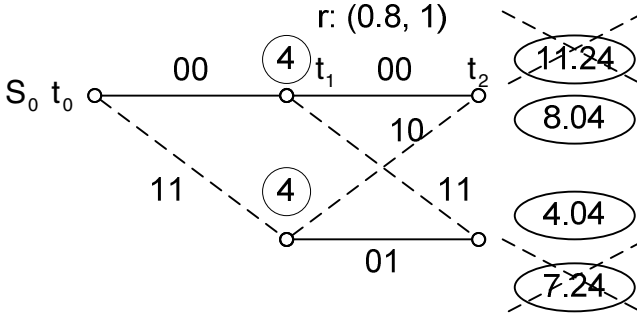
- In the second frame, the branch metrics and the path metrics are:

$$mb_2^1 = [0.8 - (-1)]^2 + [1 - (-1)]^2 = 7.24$$

$$mb_2^2 = [0.8 - 1]^2 + [1 - (+1)]^2 = 0.04$$

$$mb_2^3 = [0.8 - 1]^2 + [1 - (-1)]^2 = 4.04$$

$$mb_2^4 = [0.8 - (-1)]^2 + [1 - 1]^2 = 3.24$$



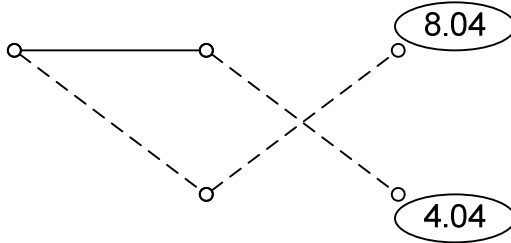
$$mp_2^1 = 4 + 7.24 = 11.24$$

$$mp_2^2 = 4 + 0.04 = 4.04$$

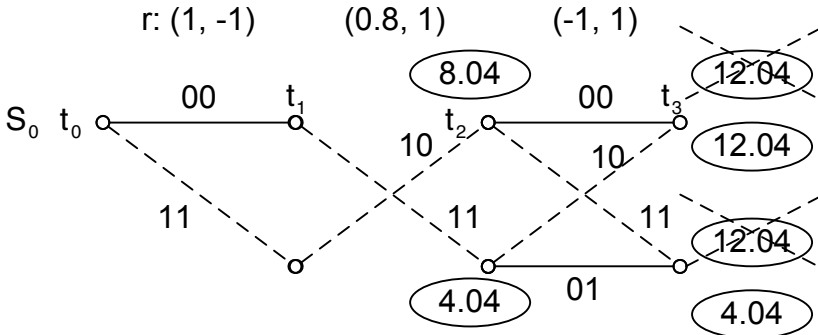
$$mp_2^3 = 4 + 4.04 = 8.04$$

$$mp_2^4 = 4 + 3.24 = 7.24$$

In each node the survivor is selected accordingly to the minimum path metric: 8.04, respectively 4.04.



– In the third frame the branch metrics and the path metrics are:



$$mb_3^1 = [-1 - (-1)]^2 + [1 - (-1)]^2 = 4$$

$$mb_3^2 = [-1 - 1]^2 + [1 - 1]^2 = 4$$

$$mb_3^3 = [-1 - 1]^2 + [1 - (-1)]^2 = 8$$

$$mb_3^4 = [-1 - (-1)]^2 + [1 - 1]^2 = 0$$

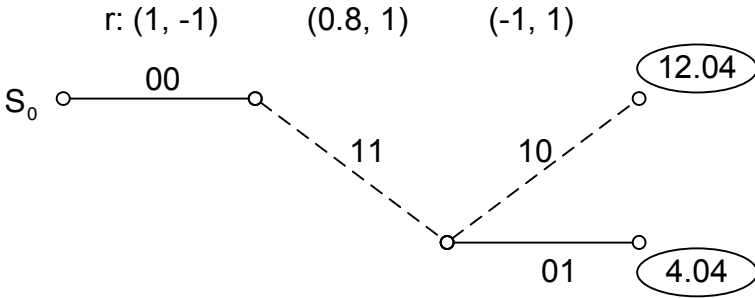
$$mp_3^1 = 8.04 + 4 = 12.04$$

$$mp_3^2 = 8.04 + 4 = 12.04$$

$$mp_3^3 = 4.04 + 8 = 12.04$$

$$mp_3^4 = 4.04 + 0 = 4.04$$

Selection of the survivors: in the first node the path metrics being the same (12.04) for the both branches, the selection is random; in the second node the survivor has the path metric 4.04. The remained ways are:



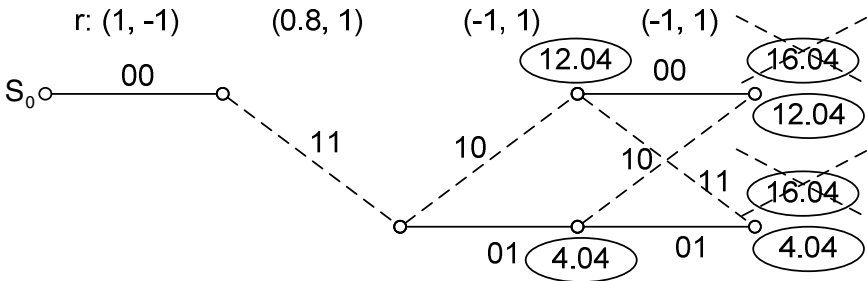
- In the fourth frame we have:

$$mb_4^1 = [-1 - (-1)]^2 + [1 - (-1)]^2 = 4$$

$$mb_4^2 = [-1 - 1]^2 + [1 - 1]^2 = 4$$

$$mb_4^3 = [-1 - 1]^2 + [1 - (-1)]^2 = 8$$

$$mb_4^4 = [-1 - (-1)]^2 + [1 - 1]^2 = 0$$



$$mp_4^1 = 12.04 + 4 = 16.04$$

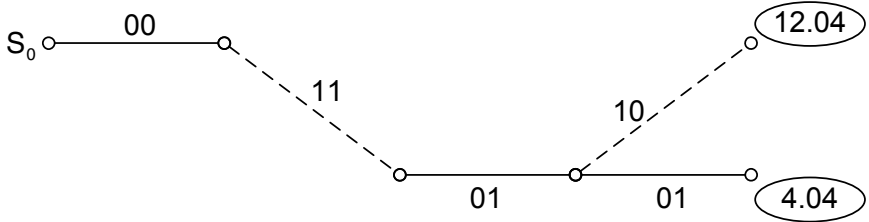
$$mp_4^2 = 12.04 + 4 = 16.04$$

$$mp_4^3 = 4.04 + 8 = 12.04$$

$$mp_4^4 = 4.04 + 0 = 4.04$$

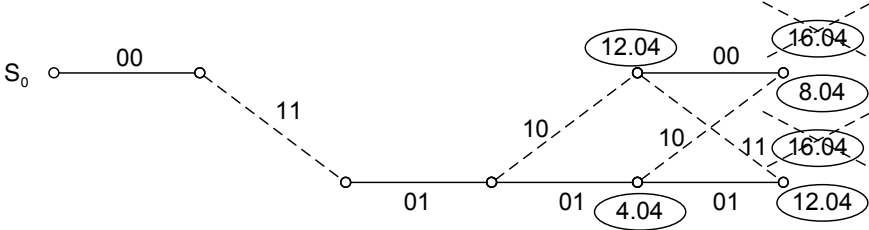
After the choice of survivor, the remaining ways are:

r: (1, -1) (0.8, 1) (-1, 1) (-1, 1)



- In the fifth frame we have:

r: (1, -1) (0.8, 1) (-1, 1) (-1, 1) (1, -1)



$$mb_5^1 = [1 - (-1)]^2 + [-1 - (-1)]^2 = 4$$

$$mb_5^2 = [1 - 1]^2 + [-1 - 1]^2 = 4$$

$$mb_5^3 = [1 - 1]^2 + [-1 - 1]^2 = 4$$

$$mb_5^4 = [1 - (-1)]^2 + [-1 - 1]^2 = 8$$

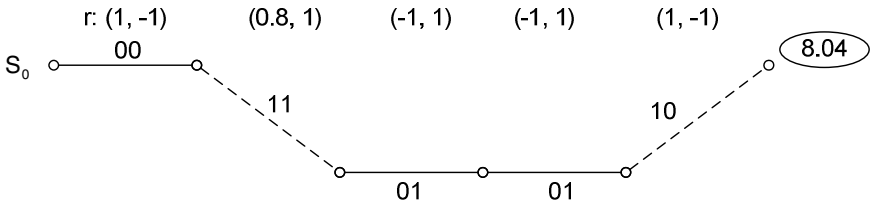
$$mp_5^1 = 12.04 + 4 = 16.04$$

$$mp_5^2 = 12.04 + 4 = 16.04$$

$$mp_5^3 = 4.04 + 4 = 8.04$$

$$mp_5^4 = 4.04 + 8 = 12.04$$

After the choice of the way with the minimum path distance, we obtain:



thus the information sequence at the output of VA decoder is:

$$\hat{i} = [0 \ 1 \ 0 \ 0 \ 1].$$

5.11.2.3 Bidirectional Soft Output Viterbi Algorithm (SOVA) [46]

The main disadvantage of VA is its output is binary (hard estimation), meaning that performance loss occurs in multistage decoding. In order to avoid this, soft outputs are required. SOVA is able to deliver soft outputs estimating for each transmitted binary symbol (i_t) the log-likelihood function $\Lambda(i_t)$. If the decision is made on finite length blocks (as for block codes), SOVA can be implemented bidirectionally, with forward and backward recursions.

- SOVA estimates the soft output information calculating the *log - likelihood function* $\Lambda(i_t)$.

$$\Lambda(i_t) = \log \frac{P(i_t = 1/r^\tau)}{P(i_t = 0/r^\tau)} \tag{5.260}$$

where $P(i_t = j/r^\tau)$ with $j=1,0$, is the a posteriori probability of the transmitted symbol, and r^τ is the received sequence of length τ .

- SOVA compares $\Lambda(i_t)$ with a zero threshold (hard decision):

$$i_t = \begin{cases} 1, & \text{if } \Lambda(i_t) \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.261}$$

- The decoder selects the path $\hat{\mathbf{x}}$ (respectively $\hat{\mathbf{i}}$) with the minimum *path metric* (mp) as the *maximum likelihood path (ML)* similar with VA.

The probability of selecting $\hat{\mathbf{x}}$ is proportional with $P(i_t/r^\tau)$, and based on relation (5.255), to $e^{-mp_{\min, \tau}}$, where $mp_{\min, \tau}$ represents the minimum path metric on τ frames. For $i_t = 1$, respectively 0 (the complementary symbol) we have.

$$P(i_t = 1/r^\tau) \sim e^{-mp_{\min, \tau}} \tag{5.262}$$

$$P(i_t = 0/r^\tau) \sim e^{-mp_{t,c}} \tag{5.263}$$

where $mp_{t,c}$ represents the minimum path metric of the paths with complementary symbol to the ML symbol at time t .

In this case, the log-likelihood function (5.260) becomes:

$$\Lambda(\mathbf{i}_t) = \log \frac{P(\mathbf{i}_t = 1/\mathbf{r}^\tau)}{P(\mathbf{i}_t = 0/\mathbf{r}^\tau)} \sim \frac{e^{-mp_{\min,\tau}}}{e^{-mp_{t,c}}} = mp_{t,c} - mp_{\min,\tau} \quad (5.264)$$

If we introduce the notations:

$$\begin{cases} mp_t^1 = mp_{\min,\tau} \\ mp_t^0 = mp_{t,c} \end{cases} \quad (5.265)$$

it follows that (5.264) can be written as:

$$\Lambda(\mathbf{i}_t) = \log \frac{P(\mathbf{i}_t = 1/\mathbf{r}^\tau)}{P(\mathbf{i}_t = 0/\mathbf{r}^\tau)} \sim mp_t^1 - mp_t^0 \quad (5.266)$$

If the ML estimate at time t is 0, we have the relations:

$$\begin{cases} mp_t^0 = mp_{\min,\tau} \\ mp_t^1 = mp_{t,c} \end{cases} \quad (5.267)$$

and log-likelihood relation becomes:

$$\begin{aligned} \Lambda(\mathbf{i}_t) &= \log \frac{P(\mathbf{i}_t = 1/\mathbf{r}^\tau)}{P(\mathbf{i}_t = 0/\mathbf{r}^\tau)} \sim \frac{e^{-mp_{t,c}}}{e^{-mp_{\min,\tau}}} = mp_{\min,\tau} - mp_{t,c} = \\ &= mp_t^0 - mp_t^1, \end{aligned} \quad (5.268)$$

meaning that the soft output of the decoder is obtained as the difference of the minimum path metric ($mp_{\min,\tau}$) among the paths having 0 information at the time t and the minimum path metric ($mp_{t,c}$) among all paths with symbol 1 at time t .

The sign of $\Lambda(\mathbf{i}_t)$ determines the hard estimate at time t and its absolute value represents the soft output information which can be used in a next stage.

If the decision is made at the end of the finite block length (as in block codes), SOVA can be implemented as a bidirectional recursive method with forward and backward recursions

Bidirectional SOVA steps:

a. Forward recursion

- 1) set the initial value: $t=0$, $S_0 = 0$, $mp_0^x(S_0 = 0) = 0$, $S_\tau = 0$ (the information block of length τ starts from S_0 and reaches 0 state at $t = \tau$; it has trellis termination)

- 2) apply Euclidian VA until the frame τ ($S_\tau = 0$)
- 3) the survivor in frame τ is the maximum likelihood path and its metric is $mp_{\min, \tau}$

b. *Backward recursion*: it applies the same steps as in forward recursion, but in opposite sense, from $t = \tau$ to $t = 0$.

- 1) set the initial values: $t = \tau$, $S_\tau = 0$, $mp_\tau^x(S_0 = 0) = 0$, $S_0 = 0$
- 2) apply Euclidian VA from $t = \tau$ to $t = 0$.

c. *Soft decision*

- 1) start at $t = 0$ and continue until $t = \tau$
- 2) at moment t , identify the ML estimate
 - $i_t = 0$ or 1
 - determine: $mp_t^i = mp_{\min, \tau}$
 - find the path metric of its best competitor, the complementary one: $i \oplus 1$:

$$mp_t^c = \min\{mp_{t-1}^f(S_k) + mb_t^c(S_k, S_j) + mp_t^b(S_j)\} \quad (5.269)$$

where S_k, S_j are the distinct states of the trellis $S_k, S_j = 0, 1, \dots, (2^{K-1} - 1)$

- $mp_{t-1}^f(S_k)$ - the path metric of the forward survivor at time $t-1$ and node S_k
- $mb_t^c(S_k, S_j)$ - the branch metric at time t for complementary symbols from node S_k to S_j
- $mp_t^b(S_j)$ - the backward survivor path metric at time t and node S_j .
 - compute

$$\Lambda(i_t) = mp_t^0 - mp_t^1 \quad (5.270)$$

Remarks concerning bidirectional SOVA

- the complexity of forward recursion is the same to that of VA.
- the complexity of backward recursion is less than VA because the survivors are not stored.

Example 5.36

Using the RSC code given in Example 5.35: $R=1/2$ and $K=2$, find the output of the decoder for the received sequence:

$$r^\tau = r^4 = [(1,1), (-1,1), (0.8,-1), (-1,-1)]$$

1. using Viterbi algorithm (VA)
2. using a bidirectional SOVA

Solution

1. The trellis of the code is given in Fig. 5.53.c and will be used for both decoding.

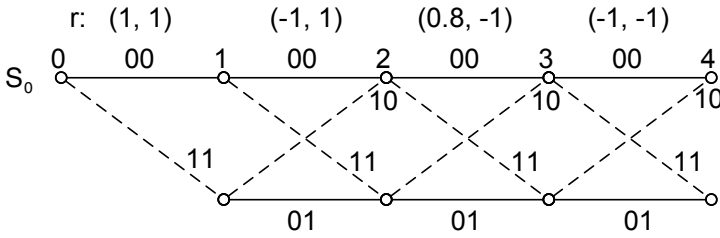
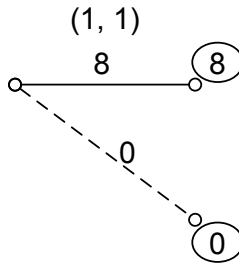


Fig. 5.54 Trellis representation on $\tau=4$ frames of RSC code with $R=1/2$ and $K=2$

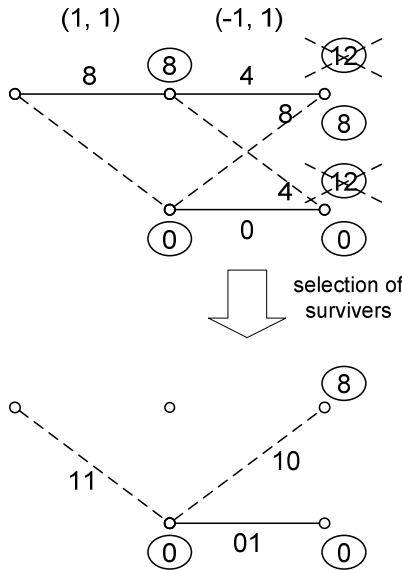
- In the first frame: $t=1$



$$mb_1^1 = [1 - (-1)]^2 + [1 - (-1)]^2 = 8 = mp_1^1$$

$$mb_1^2 = [1 - 1]^2 + [1 - 1]^2 = 0 = mp_1^2$$

- In the second frame: $t=2$



$$mb_2^1 = [-1 - (-1)]^2 + [1 - (-1)]^2 = 4$$

$$mp_2^1 = 8 + 4 = 12$$

$$mb_2^2 = [-1 - 1]^2 + [1 - (-1)]^2 = 8$$

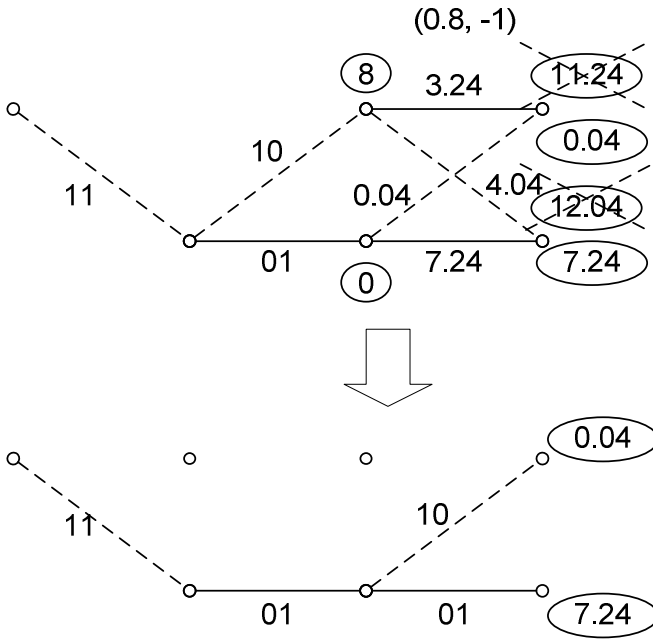
$$mp_2^2 = 0 + 8 = 18$$

$$mb_2^3 = [-1 - 1]^2 + [1 - 1]^2 = 4$$

$$mp_2^3 = 8 + 4 = 12$$

$$mb_2^4 = [-1 - (-1)]^2 + [1 - 1]^2 = 0$$

- In the third frame: $t=3$



$$mb_3^1 = [0.8 - (-1)]^2 + [-1 - (-1)]^2 = 3.24$$

$$mp_3^1 = 8 + 3.24 = 11.24$$

$$mb_3^2 = [0.8 - 1]^2 + [-1 - (-1)]^2 = 0.04$$

$$mp_3^2 = 0 + 0.04 = 0.04$$

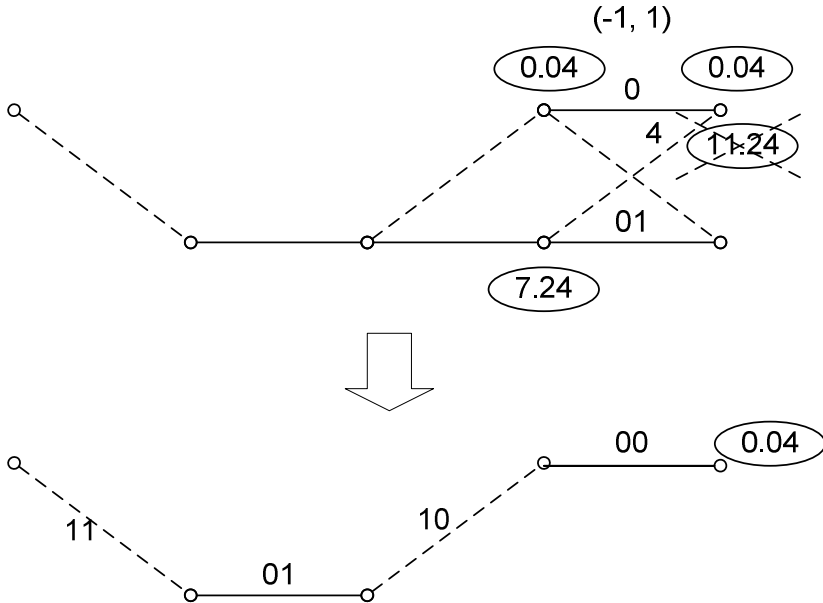
$$mb_3^3 = [0.8 - 1]^2 + [-1 - 1]^2 = 4.04$$

$$mp_3^3 = 8 + 4.04 = 12.04$$

$$mb_3^4 = [0.8 - (-1)]^2 + [-1 - 1]^2 = 7.24$$

$$mp_3^4 = 0 + 7.24 = 7.24$$

- In the fourth frame: $t=4$



$$mb_4^1 = [-1 - (-1)]^2 + [1 - (-1)]^2 = 0$$

$$mp_4^1 = 0.04 + 4 = 4.04$$

$$mb_4^2 = [-1 - 1]^2 + [-1 - (-1)]^2 = 4$$

$$mp_4^2 = 7.24 + 4 = 11.24$$

The ML path, with the minimum path metric is: $\hat{v} = [11101110100]$ corresponding to the estimation:

$$\hat{i} = [1 \ 0 \ 1 \ 0]$$

2. SOVA

- Forward recursion is obtained applying VA starting from $S_0 = 0$ until $S_\tau = 4$

The result of VA for the forward recursion is presented in Fig. 5.55.

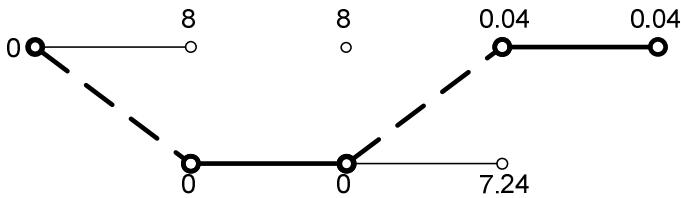


Fig. 5.55 Forward recursion; the thick line is representing the ML path (with minimum path metric 0.04)

The survivor path metrics are indicated above each node. The final survivor is the ML path with the metric $mp_{\min, \tau=4} = 0.04$.

b. *Backward recursion*

VA is applied starting with $t = \tau = 4$, until $t = 0$, in reverse sense. The backward recursion is shown in Fig. 5.56. The first number above each node indicates the forward survivor path metric and the second one the backward path metrics.

$$r: (1,1) (-1,1) (0.8,-1) (-1,-1)$$

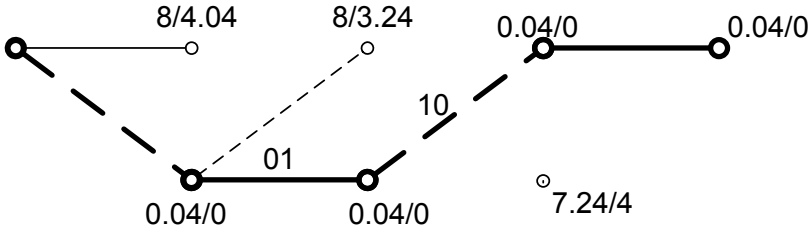


Fig. 5.56 The backward recursion; ML path is presented with thick line.

Remark: we propose to the reader to apply VA in the reverse sense and to check the results with the ones given in Fig. 5.56.

c. *Soft decision*

- We start at $t = 0$
- At moment $t = 1$, we identify the ML estimate (hard estimate): $i_1 = 1$ and thus $mp_1^1 = mp_{\min,4} = 0.04$

The complementary path metric is mp_1^0 and is the minimum path metric of the path that have 0 at time $t = 1$; its path metrics is calculated from Fig. 5.56 as the sum of the forward and backward survivor path metrics at node 0:

$$mp_1^0 = 8 + 4.04 = 12.04$$

Accordingly to (5.270), the log-likelihood ratio at time $t = 1$ is:

$$\Lambda(i_1) = mp_1^0 - mp_1^1 = 12.04 - 0.04 = 12$$

- at moment $t = 2$, the ML estimate is $i_2 = 0$ and the minimum path metric is $mp_2^0 = mp_{\min,4} = 0.04$

- the best complementary competitor path at time $t = 2$, noted mp_2^1 , is according to (5.269):

$$mp_2^1 = \min_{S_k, S_j} \{ mp_1^f(S_k) + mb_2^1(S_k, S_j) + mp_2^b(S_j) \}$$

where $S_i, S_j = \{0,1\}$, $mp_1^f(S_k)$ is the forward survivor path metric at time 1 and node $S_k = \{1, 0\}$, $mb_2^1(S_k, S_j)$ is the branch metric at time $t = 2$ for the input information 1 and $mp_2^b(S_j)$ is the backward survivor path metric at moment $t = 2$ and node $S_j = \{0,1\}$

$$mp_2^1 = \min\{(0 + 8 + 3.24), (8 + 4 + 0.04)\} = 11.24$$

- the log-likelihood ratio at time $t = 2$ is:

$$\Lambda(i_2) = mp_2^0 - mp_2^1 = 0.04 - 11.24 = -11.2$$

In the same way we obtain:

- at $t = 3$:

$$mp_3^1 = \min\{(8 + 4.04 + 4), (0 + 0.04 + 0)\} = 0.04$$

$$mp_3^0 = \min\{(0 + 7.24 + 4), (8 + 3.24 + 0)\} = 11.24$$

$$\Lambda(i_3) = mp_3^0 - mp_3^1 = 11.24 - 0.04 = 11.2$$

- at $t = 4$:

$$mp_4^0 = \min\{(0.04 + 0 + 0)\} = 0.04$$

$$mp_4^1 = \min\{(7.24 + 4 + 0)\} = 11.24$$

$$\Lambda(i_4) = mp_4^0 - mp_4^1 = 0.04 - 11.24 = -11.2$$

Thus, comparing $\Lambda(i_t)$ with zero threshold, the estimated analogue outputs are: $\{12, -12.2, 11.2, -11.2\}$, corresponding to the digital outputs: $\{1\ 0\ 1\ 0\}$.

We check easily that, if hard decision would be used, the decoded sequence is the same with the one obtained previously using VA: $\hat{f} = [1\ 0\ 1\ 0]$.

5.11.2.4 MAP Algorithm

MAP is an optimal decoding algorithm which minimizes the symbol or bit error probability. It computes the log-likelihood ratio (5.260), which is a soft information and can be used in further decoding stages. This value is used to generate a hard estimate using a comparison with zero threshold as in relation (5.261).

We intend to illustrate MAP decoding using an example.

Example 5.37

MAP decoding for a RSC code with $R = 1/2$ and $K = 3$

The block-scheme of the system is presented in Fig. 5.57.

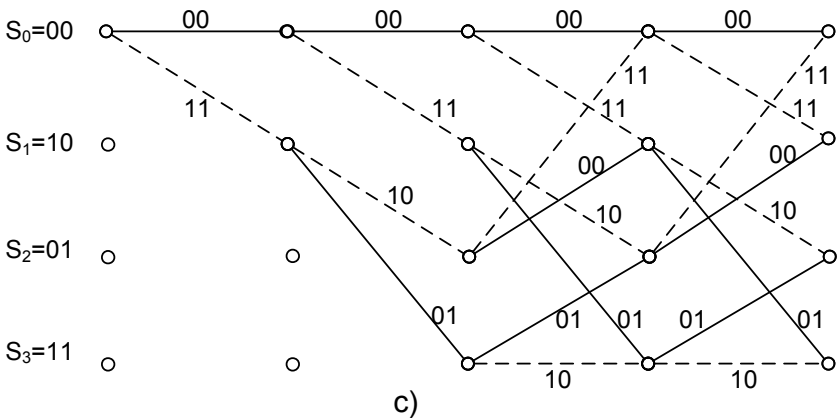
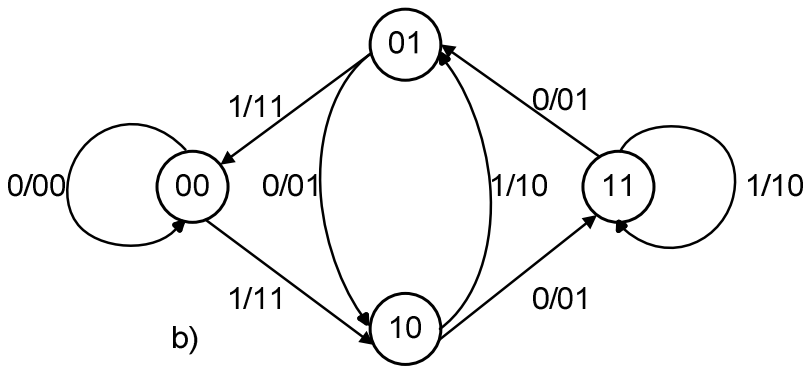
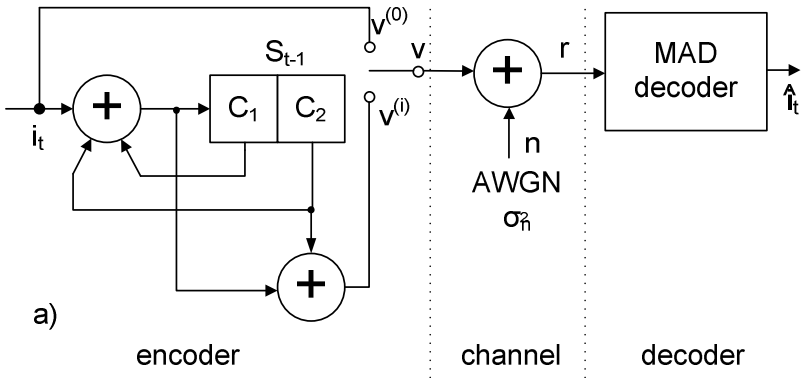


Fig. 5.57 MAP decoding illustration: a) block scheme of a transmission system with RSC ($R = 1/2$, $K = 3$, $G = [1, (1 + x^2)/(1 + x + x^2)]$) and MAP decoding; b) state transition diagram of RSC encoder with $R = 1/2$, $K = 3$, $G = [1, (1 + x^2)/(1 + x + x^2)]$; c) trellis diagram of RSC code from b).

We assume that the generated bits $\mathbf{v}^{(0)}, \mathbf{v}^{(1)}$ are BPSK modulated ($x^{(0)} = -1, x^{(1)} = +1$) before entering the channel (AWGN with dispersion σ_n^2).

The decoding is block mode, meaning that decoding starts after a complete block of length τ receiving (\mathbf{r}^τ) . It is assumed that decoding starts from all zero state of the encoder (\mathbf{S}_0) and at the end of decoding (after τ frames), it reaches the same state \mathbf{S}_0 .

The transition probabilities of the AWGN channel are:

$$P(\mathbf{r}^\tau / \mathbf{x}^\tau) = \prod_{j=1}^{\tau} P(r_j / x_j) \quad (5.271)$$

$$P(\mathbf{r}_j / \mathbf{x}_j) = \prod_{k=0}^{n-1} P(r_{j,k} / x_{j,k}) \quad (5.272)$$

$$P(r_{j,k} / x_{j,k} = -1) = \frac{1}{\sqrt{2\pi\sigma_n^2}} e^{-\frac{(r_{j,k}+1)^2}{2\sigma_n^2}} \quad (5.273)$$

$$P(r_{j,k} / x_{j,k} = +1) = \frac{1}{\sqrt{2\pi\sigma_n^2}} e^{-\frac{(r_{j,k}-1)^2}{2\sigma_n^2}} \quad (5.274)$$

Be i_t the information bit associated with the transition S_{t-1} to S_t and producing the output \mathbf{v}_t . The decoder will give an estimate \hat{i}_t , examining \mathbf{r}_t^τ .

The MAP algorithm calculates for each i_t the log likelihood ratio ($\Lambda(i_t)$) based on the received sequence \mathbf{r}^τ , according to the relation (5.260). MAP decoder makes a decision comparing $\Lambda(i_t)$ to a threshold equal to zero (relation (5.261)).

The aposteriori probabilities (APP) can be calculated using Bayes relations (5.19):

$$\begin{aligned} P(\hat{i}_t = 0 / \mathbf{r}^\tau) &= \sum_{(l',1) \in \mathbf{T}_t^0} P(\mathbf{S}_{t-1} = l', \mathbf{S}_t = 1 / \mathbf{r}^\tau) = \\ &= \sum_{(l',1) \in \mathbf{T}_t^0} \frac{P(\mathbf{S}_{t-1} = l', \mathbf{S}_t = 1, \mathbf{r}^\tau)}{P(\mathbf{r}^\tau)} \end{aligned} \quad (5.275)$$

where \mathbf{T}_t^0 is the set of transitions $\mathbf{S}_{t-1} = l'$ to $\mathbf{S}_t = 1$ that are caused by information bit $i_t = 0$.

Analogue is calculated the APP of the information $i_t = 1$:

$$\begin{aligned} P(i_t = 1 / \mathbf{r}^\tau) &= \sum_{(l',1) \in \mathbf{T}_t^1} P(\mathbf{S}_{t-1} = l', \mathbf{S}_t = 1, \mathbf{r}^\tau) = \\ &= \sum_{(l',1) \in \mathbf{T}_t^1} \frac{P(\mathbf{S}_{t-1} = l', \mathbf{S}_t = 1, \mathbf{r}^\tau)}{P(\mathbf{r}^\tau)} \end{aligned} \quad (5.276)$$

where \mathbf{T}_t^1 is the set of transitions $\mathbf{S}_{t-1} = l'$ to $\mathbf{S}_t = 1$ that are caused by information bit $i_t = 1$.

The joint probabilities will be noted with

$$\sigma_t(l', 1) = P(\mathbf{S}_{t-1} = l', \mathbf{S}_t = 1, \mathbf{r}^\tau) \quad (5.277)$$

and (5.275) and (5.276) will be:

$$P(i_t = 0 / \mathbf{r}^\tau) = \sum_{(l',1) \in \mathbf{T}_t^0} \frac{\sigma_t(l', 1)}{P(\mathbf{r}^\tau)} \quad (5.275a)$$

$$P(i_t = 1 / \mathbf{r}^\tau) = \sum_{(l',1) \in \mathbf{T}_t^1} \frac{\sigma_t(l', 1)}{P(\mathbf{r}^\tau)} \quad (5.276a)$$

The log-likelihood ratio can be expressed as:

$$\Lambda(i_t) = \log \frac{\sum_{(l',1) \in \mathbf{T}_t^1} \sigma_t(l', 1)}{\sum_{(l',1) \in \mathbf{T}_t^0} \sigma_t(l', 1)} \quad (5.278)$$

The calculation of joint probabilities $\sigma_t(l', 1)$ can be done introducing the following probabilities:

$$\alpha_t(1) = P(\mathbf{S}_t = 1, \mathbf{r}^\tau) \quad (5.279)$$

$$\beta_t(1) = P(\mathbf{r}_{t+1}^\tau / \mathbf{S}_t = 1) \quad (5.280)$$

$$\gamma_t^i(1) = P(i_t = i, \mathbf{S}_t = 1, \mathbf{r}^\tau / \mathbf{S}_{t-1} = l'), i = \{0, 1\} \quad (5.281)$$

According to these notations, σ_t and $\Lambda(i_t)$ can be written:

$$\sigma_t(l', 1) = \alpha_{t-1}(l') \beta_t(1) \sum_{i=0,1} \gamma_t^i(l', 1) \quad (5.282)$$

$$\Lambda(i_t) = \log \sum_{(l',1) \in \mathbf{T}_t^1} \frac{\alpha_{t-1}(l') \gamma_t^1(l', 1) \beta_t(1)}{\alpha_{t-1}(l') \gamma_t^0(l', 1) \beta_t(1)} \quad (5.283)$$

and also:

$$\alpha_i(l) = \sum_{l' \in \mathbf{S}} \alpha_{t-1}(l') \sum_{\mathbf{i}=\{0,1\}} \gamma_t^{\mathbf{i}}(l', l) \quad (5.284)$$

$$\beta_i(l) = \sum_{l' \in \mathbf{S}} \beta_{t+1}(l') \sum_{\mathbf{i}=\{0,1\}} \gamma_{t+1}^{\mathbf{i}}(l', l) \quad (5.285)$$

$$\gamma_t^{\mathbf{i}}(l', l) = \begin{cases} P_t(\mathbf{i}) \exp \left[-\frac{\sum_{j=0}^{n-1} [r_{t,j}^{n-1} - x_{t,j}^{\mathbf{i}}(l)]^2}{2\sigma_n^2} \right], & \text{for } (l, l') \in \mathbf{T}_t^{\mathbf{i}} \\ 0, & \text{otherwise} \end{cases} \quad (5.286)$$

$P_t(\mathbf{i})$ is the a priori probability of information bit at moment t : $i_t = i$, and $x_t^{\mathbf{i}}(l)$ is the output of the encoder (after BPSK modulation), associated with transition $\mathbf{S}_{t-1} = l'$ to $\mathbf{S}_t = l$ and input $i_t = i$. \mathbf{S} is representing the distinct states of the encoder (in number of 2^{K-1} , noted from 0 to $2^{K-1}-1$).

In summary, MAP algorithm has the followings routines:

1. *Forward recursion* (Fig. 5.58.a)

- initialization of $\alpha(l)$ with $l \in \mathbf{S}$

$$\mathbf{S} = \{0, 1, \dots, 2^{K-1} - 1\},$$

$$\alpha_0(0) = 1 \text{ and } \alpha_0(l) = 0 \text{ for } l \neq 0.$$

- for $t = \overline{1, \tau}$ and $l \in \mathbf{S}$ calculate for all l branches of the trellis:

$$\gamma_t^{\mathbf{i}}(l', l) = P_t(\mathbf{i}) \exp \left[\frac{-d^2(\mathbf{r}_t, x_t)}{2\sigma_n^2} \right], \mathbf{i} = \{0, 1\} \quad (5.285a)$$

where $P_t(\mathbf{i})$ is the a priori probabilities of each information bit (in most cases the equally probability is assumed), and $d^2(\mathbf{r}_t, x_t)$ is the squared Euclidean distance between \mathbf{r}_t and x_t (the modulated i_t).

- for $\mathbf{i} = \{0, 1\}$, store $\gamma_t^{\mathbf{i}}(l', l)$
- for $t = \overline{1, \tau}$ and $l \in \mathbf{S}$ calculated and store $\alpha_t(l)$ given by (5.284).

A representation of forward recursion is given in Fig. 5.57.a.

2. *Backward recursion* (Fig. 5.57.b)

- initialization of $\beta_\tau(l)$, $l \in \mathbf{S}$

$$\beta_\tau(0) = 1 \text{ and } \beta_\tau(l) = 0 \text{ for } l \neq 0$$

- for $t = \overline{1, \tau}$ and $l \in \mathbf{S}$, calculate $\beta_\tau(l)$ with (5.285)
- for $t < \tau$ calculate the log-likelihood $\Lambda(i_t)$ with (5.283)

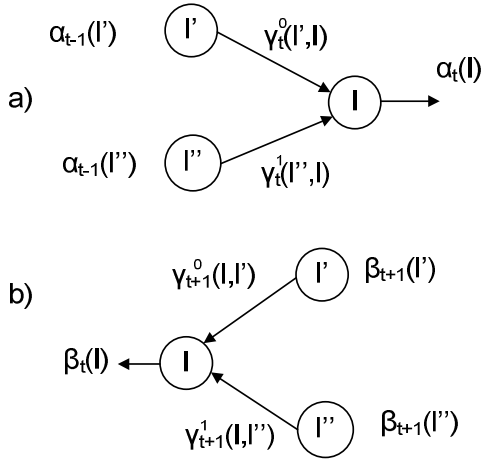


Fig. 5.58 Graphical MAP representation: a) forward recursion, b) backward recursion

Example 5.38

Be the RSC code from Example 5.35 ($R = \frac{1}{2}, G = \left[1, \frac{1}{1+x} \right]$), with BPSK modulation and AWGN channel with $\frac{E_b}{N_0} = 2\text{dB}$. The information sequence is $i = (1\ 1\ 0\ 0\ 1)$.

After $\tau = 5$, the received sequence is:

$$r^{(0)} = 0,030041 \quad -0,570849 \quad -0,38405 \quad -0,744790 \quad 0,525812 \quad 0,507154$$

$$r^{(1)} = 0,726249 \quad -0,753015 \quad -1,107597 \quad -0,495092 \quad 1,904994 \quad -1,591323$$

The power of the signal is $P_s = 0,5\text{ W}$.

Find the soft and hard outputs of a MAP decoder.

Solution

The encoder is presented in Fig. 5.53a and the state-diagram, respectively the trellis are given in Fig. 5.53b, respectively c.

In order to apply MAP, we need the trellis, the received sequence and also the knowledge of channel dispersion σ_n^2 .

The signal per noise ratio (SNR) can be expressed as:

$$\frac{S}{N} = \frac{P_S}{P_N} = R \frac{E_b}{N_0} \quad (5.287)$$

for BPSK modulation, where R is the coding rate. Taking into account that the length of the sequence $L = \tau = 5$ is small compared to the constant length ($K = 2$), R is expressed using (5.213):

$$R = \frac{mL}{nL + K} = \frac{1 \cdot 5}{2 \cdot 5 + 2} = 0,416$$

It follows from (5.287) that:

$$P_N = \sigma_n^2 = \frac{P_S}{R \cdot E_b / N_0} = \frac{0,5}{0,416 \cdot 10^{0,2}} = 0,758$$

Then, MAP steps are applied:

1. forward recursion

According to the initialization, we have:

$$\alpha_0(0) = 1 \text{ and } \alpha_0(1) = 0 \text{ for } l \neq 0$$

For $t = \overline{1, 5}$, according to (5.284) we have:

$$\alpha_t^i(1) = \sum_{l=0}^1 \alpha_{t-1}(l') \sum_{i=\{0,1\}} \gamma_t^i(l', 1)$$

– for $t = 1$, we have

$$\alpha_1(0) = \alpha_0(0) [\gamma_1^0(0, 0) + \gamma_1^1(0, 0)] + \alpha_0(1) [\gamma_1^0(1, 0) + \gamma_1^1(1, 0)]$$

We do not have $\gamma_1^1(0, 0)$ and $\gamma_1^0(1, 0)$, because in our case \mathbf{T}_t^0 consists of $(0, 0)$ and $(1, 1)$ (if at the input is “0” we can have transitions only from $l' = 0$ to $l = 0$ or from $l' = 1$ to $l = 1$).

For \mathbf{T}_t^1 it consists of $(1, 0)$ and $(0, 1)$, if at the input we have “1” there are transitions from $l' = 1$ to $l = 0$ or $l' = 0$ to $l = 1$.

$$\alpha_1(1) = \alpha_0(0) [\gamma_1^0(0, 1) + \gamma_1^1(0, 1)] + \alpha_0(1) [\gamma_1^0(1, 1) + \gamma_1^1(1, 1)]$$

According to the above explanation, we obtain:

$$\begin{cases} \alpha_t(0) = \alpha_{t-1}(0) \gamma_t^0(0, 0) + \alpha_{t-1}(1) \gamma_t^1(1, 0) \\ \alpha_t(1) = \alpha_{t-1}(0) \gamma_t^1(0, 1) + \alpha_{t-1}(1) \gamma_t^0(1, 1) \end{cases}$$

– for $t = \overline{1, 6} \Rightarrow$

$$\begin{cases} \alpha_1(0) = \alpha_0(0)\gamma_1^0(0, 0) + \alpha_0(1)\gamma_1^1(1, 0) = 1 \cdot 0,034678 + 0 = 0,034678 \\ \alpha_1(1) = \alpha_0(0)\gamma_1^0(0, 1) + \alpha_0(1)\gamma_1^1(1, 1) = 1 \cdot 0,255655 + 0 = 0,255655 \\ \alpha_2(0) = \alpha_1(0)\gamma_2^0(0, 0) + \alpha_1(1)\gamma_2^1(1, 0) = 0,034678 \cdot 0,425261 + \\ \quad + 0,255655 \cdot 0,094143 = 0,038815 \\ \alpha_2(1) = \alpha_1(0)\gamma_2^0(0, 1) + \alpha_1(1)\gamma_2^1(1, 1) = 0,015322 \end{cases}$$

and so on, until:

$$\begin{cases} \alpha_5(0) = \alpha_4(0)\gamma_5^0(0, 0) + \alpha_4(1)\gamma_5^1(1, 0) = 0,00000338 \\ \alpha_5(1) = 0,001770 \\ \alpha_6(0) = \alpha_5(0)\gamma_6^0(0, 0) + \alpha_5(1)\gamma_6^1(1, 0) = 0,000599 \\ \alpha_6(1) = \alpha_5(0)\gamma_6^0(0, 1) + \alpha_5(1)\gamma_6^1(1, 1) = 0,000002356 \end{cases}$$

2. backward recursion, applying relation (5.285)

$$\beta_t(l) = \sum_{l'=0}^1 \beta_{t+1}(l') \sum_{i=\{0,1\}} \gamma_{t+1}^i(l', 1)$$

$\beta_\tau(0) = 1$ and $\beta_\tau(l) = 0$ for $l \neq 0$ (initialization)

In our example:

$$\begin{aligned} \beta_t(l=0) &= \beta_t(0) = \beta_{t+1}(0) \left[\gamma_{t+1}^0(0, 0) + \gamma_{t+1}^1(0, 0) \right] + \\ &\quad + \beta_{t+1}(1) \left[\gamma_{t+1}^0(0, 1) + \gamma_{t+1}^1(0, 1) \right] = \\ &= \beta_{t+1}(0) \gamma_{t+1}^0(0, 0) + \beta_{t+1}(1) \gamma_{t+1}^1(0, 1) \\ \beta_t(l) &= \beta_{t+1}(0) \left[\gamma_{t+1}^0(1, 0) + \gamma_{t+1}^1(1, 0) \right] + \beta_{t+1}(1) \left[\gamma_{t+1}^0(1, 1) + \gamma_{t+1}^1(1, 1) \right] = \\ &= \beta_{t+1}(0) \gamma_{t+1}^1(1, 0) + \beta_{t+1}(1) \gamma_{t+1}^0(1, 1) \\ \left. \begin{cases} \beta_6(0) = 1 \\ \beta_6(1) = 0 \end{cases} \right\} &\text{initialization} \\ \left\{ \begin{aligned} \beta_5(0) &= \beta_6(0)\gamma_6^0(0, 0) + \beta_6(1)\gamma_6^1(0, 1) = 1 \cdot 0,088558 + 0 = 0,088558 \\ \beta_5(1) &= \beta_6(0)\gamma_6^1(1, 0) + \beta_6(1)\gamma_6^0(1, 1) = 1 \cdot 0,338085 + 0 = 0,338085 \end{aligned} \right. \end{aligned}$$

and so, until

$$\left\{ \begin{aligned} \beta_1(0) &= \beta_2(0)\gamma_2^0(0, 0) + \beta_2(1)\gamma_2^1(0, 1) = 0,005783 \\ \beta_1(1) &= \beta_2(0)\gamma_2^1(1, 0) + \beta_2(1)\gamma_2^0(1, 1) = 0,001557 \end{aligned} \right.$$

With α_t, β_t above calculated, we can calculate log-likelihood ratio according with (5.283)

$$\Lambda(i_t) = \log \frac{\alpha_{t-1}(0)\gamma_t^1(0,1)\beta_t(1) + \alpha_{t-1}(1)\gamma_t^1(1,0)\beta_t(0)}{\alpha_{t-1}(0)\gamma_t^0(0,0)\beta_t(0) + \alpha_{t-1}(1)\gamma_t^0(1,1)\beta_t(1)}$$

$$\Lambda(i_1) = \ln \frac{\alpha_0(0)\gamma_1^1(0,1)\beta_1(1) + \alpha_0(1)\gamma_1^1(1,0)\beta_1(0)}{\alpha_0(0)\gamma_1^0(0,0)\beta_1(0) + \alpha_0(1)\gamma_1^0(1,1)\beta_1(1)} \approx 0,685564$$

and so on, until

$$\Lambda(i_6) = \ln \frac{\alpha_5(0)\gamma_6^1(0,1)\beta_6(1) + \alpha_5(1)\gamma_6^1(1,0)\beta_6(0)}{\alpha_5(0)\gamma_6^0(0,0)\beta_6(0) + \alpha_5(1)\gamma_6^0(1,1)\beta_6(1)} \approx 7,72990$$

Comparing the soft outputs $\Lambda(i_t)$ to the threshold 0, we obtain the hard estimates:

$$\hat{i} = (11001).$$

5.11.2.5 MAX-LOG-MAP Algorithm

MAP decoding algorithm is considered too complex for implementation in many applications because of the required large memory and of the large time necessary to calculate a numerous multiplications and exponentials.

A solution to avoid these drawbacks of MAP is to use the logarithms of $\gamma_t^i(l',1), \alpha_t(l), \beta_t(l)$:

$$\log \gamma_t^i(l',1) = \overline{\gamma}_t^i(l',1) \quad (5.288)$$

$$\log \alpha_t(l) = \overline{\alpha}_t(l) = \log \sum_{l' \in \mathbf{S}} \sum_{\mathbf{i}=\{0,1\}} e^{\overline{\alpha}_{t-1}(l') + \overline{\gamma}_t^i(l',1)} \quad (5.289)$$

with initial conditions $\overline{\alpha}_0(0) = 0$ and $\overline{\alpha}_0(1) = -\infty$, if $l \neq 0$.

$$\log \beta_t(l) = \overline{\beta}_t(l) = \log \sum_{l' \in \mathbf{S}} \sum_{\mathbf{i}=\{0,1\}} e^{\overline{\beta}_{t+1}(l) + \overline{\gamma}_{t+1}^i(l',1)} \quad (5.290)$$

with initial conditions $\overline{\beta}_\tau(0) = 0$ and $\overline{\beta}_\tau(1) = -\infty$, if $l \neq 0$.

The relation (5.283) can be written as:

$$\Lambda(\hat{i}_t) = \log \frac{\sum_{l' \in \mathbf{S}} e^{\overline{\alpha}_{t-1}(l') + \overline{\gamma}_t^1(l',1) + \overline{\beta}_t(1)}}{\sum_{l' \in \mathbf{S}} e^{\overline{\alpha}_{t-1}(l') + \overline{\gamma}_t^0(l',1) + \overline{\beta}_t(1)}} \quad (5.291)$$

The above expression can be simplified using the approximation

$$\log(e^{x_1} + e^{x_2} + \dots + e^{x_n}) \approx \max_{i=1,n} x_i \quad (5.292)$$

The relation (5.291) can be approximated by:

$$\Lambda(\mathbf{i}_t) \cong \max_{l \in \mathbf{S}} \left[\overline{\alpha}_{t-1}(l') + \overline{\gamma}_t^1(l', 1) + \overline{\beta}_t(1) \right] - \max_{l \in \mathbf{S}} \left[\overline{\alpha}_{t-1}(l') + \overline{\gamma}_t^0(l', 1) + \overline{\beta}_t(1) \right] \quad (5.293)$$

The computation of $\overline{\alpha}_t(l')$ and $\overline{\beta}_t(1)$ in (5.293) is equivalent to the computation of forward and backward recursion from Viterbi algorithm, since they can be written as:

$$\overline{\alpha}_t(1) = \max_{l \in \mathbf{S}} \left[\overline{\alpha}_{t-1}(l') + \overline{\gamma}_t^1(l', 1) \right], l \in \mathbf{S}, i = \{0, 1\} \quad (5.294)$$

$$\overline{\beta}_t(1) = \max_{l \in \mathbf{S}} \left[\overline{\beta}_{t+1}(l') + \overline{\gamma}_{t+1}^1(l', 1) \right], l', 1 \in \mathbf{S}, i = \{0, 1\} \quad (5.295)$$

For each bit, the MAX-LOG-MAP algorithm calculates two Viterbi metrics and takes the largest one.

5.11.2.6 LOG-MAP Algorithm

Using the approximation relation (5.293), MAX-LOG-MAP algorithm is not anymore optimal, as MAP it is. An improvement can be obtained if Jacobian algorithm [13] is used.

It is based on a better approximation of logarithm of a sum of exponentials:

$$\begin{aligned} \log(e^{z_1} + e^{z_2}) &= \max(z_1, z_2) + \log\left(1 + e^{-|z_2 - z_1|}\right) = \\ &= \max(z_1, z_2) + \text{fc}(|z_2 - z_1|) \end{aligned} \quad (5.296)$$

where $\text{fc}(|z_2 - z_1|)$ is a correction function, that can be implemented using look-up table.

For a sum of n exponentials, the approximation relation (5.296) can be computed by a recursive relation:

$$\log(e^{z_1} + \dots + e^{z_n}) = \log(\Delta + e^{z_n}) \quad (5.297)$$

with

$$\Delta = e^{z_1} + \dots + e^{z_{n-1}} = e^z \quad (5.298)$$

such that:

$$\begin{aligned} \log(e^{z_1} + \dots + e^{z_n}) &= \max(\log \Delta, z_n) + \text{fc}(|\log \Delta - z_n|) = \\ &= \max(\log z, z_n) + \text{fc}(|\log z - z_n|) \end{aligned} \quad (5.299)$$

Such a recursive relation (5.299) can be used to evaluate $\Lambda(i_t)$ in relation (5.293). We identify z_n :

$$z_n = \overline{\alpha}_{t-1}(n') + \gamma_t^i(n', n) + \overline{\beta}_t(n), \text{ where } n', n \text{ are } l', l \in \mathbf{S} \text{ and } i = \{0, 1\} \quad (5.300)$$

The performance of LOG-MAP is the same with MAP algorithm, but the complexity is higher than that of MAX-LOG-MAP [46].

5.11.2.7 Comparison of Decoding Algorithms

A very good comparison of all decoding algorithms is made in [46]. Some of these ideas will be presented in what follows:

- MAP, the optimal decoding algorithm, in order to calculate $\Lambda(i_t)$, for each i_t , considers all paths in the trellis, divided in two complementation sets (one that has $i_t = 1$ and the second which has $i_t = 0$).
- MAX-LOG-MAP considers only two paths for each i_t : the best path with $i_t = 0$ and the best path with $i_t = 1$. It computes $\Lambda(i_t)$ for each paths and returns its difference.
- SOVA takes two paths for each i_t : one is the ML path and the other one is the best path with $\overline{i_t}$ (complementary i_t) to ML path, meaning that these two paths are identical to those considered by MAX-LOG-MAP.

A comparison of complexity between all these algorithms is given in [46].

Remarks

- the performance of soft output algorithms cannot be done in only one iteration of decoding (BER); BER is defined only for hard estimation;
- in [46], [21] are given performance comparison of distinct decoding algorithms.

5.12 Line (baseband) Codes

Baseband (BB) is defined as the bandwidth of the message signal, without frequency transform, whereas the radiofrequency (RF) bandwidth refers to the bandwidth of the band pass modulation.

Baseband codes are called also *line codes*, *transmission modes*, *baseband formats/wave formats*. These codes were developed for digital transmission over telephone cables (coaxial twisted pairs) or digital recording on magnetic media (in the 60's) and recently for optic fiber transmission systems. In this category are included also the codes used on free space optics (infrared remote control), codebase for paper printing, magnetic and optic recording, but these one will not be presented on the present book.

Line codes act as clothes that information wears in transmission or storage in different media (cables, wireless, optic fiber, magnetic tape, CD, DVD, etc.)

The choice of a line code from a large family (line codes wardrobe) is done taking into account several aspects, which need to be appropriate to the application (it is analog to put a beautiful dressing for a wedding and not to take a swim suit):

- *signal spectrum* (in fact the PSD - power spectral density): are important the bandwidth and the absence of the DC component.

The presence of DC component requires DC couplings, meanwhile its absence allows AC couplings (transformers), the latter one ensures electrical isolation, this minimizing interferences. In storage, for magnetic recordings, the absence of DC component is also desired.

The bandwidth of line codes need to be small compared to channel bandwidth to avoid ISI (intersymbol interference –see 2.8.5).

PSD of distinct line codes is obtained as Fourier transform of the autocorrelation function $R(\tau)$, and their graphical representation is given in [31], [39] and [49].

A coarse estimation of the bandwidth can be done using relation (2.32):

$$\frac{1}{T} = \overset{\bullet}{M} = 1,25B$$

it follows that narrow pulses in the encoded sequence require larger bandwidth and vice versa.

- *bit or symbol synchronization* (timing) is done using the received signal, based on signal transitions from high to low or low to high levels, meaning that line code format providing enough transition density in the coded sequence will be preferred.
- *code efficiency* [49], η is defined as:

$$\eta =: \frac{\text{actual information rate}}{\text{theoretical maximum information rate}} [x 100] \quad (5.301)$$

- *ratio* $\overset{\bullet}{D}/\overset{\bullet}{M} \left(\frac{\text{decision rate}}{\text{signalling speed}} \right)$; certain codes provide an increase of $\overset{\bullet}{D}$ for the same signalling speed, so without increasing the frequency bandwidth of the signal
- *error performance*: BER expressed as a function of SNR
- *error detection capacity*: without being error control codes (as presented in 5), certain BB codes have features that can be used in error detection.
- *immunity to polarity inversion*: same BB codes, the differential ones, are immune to the accidental inversion of the wires in a pair, causing the inversion of “0” and “1”
- *complexity and costs* of the encoding-decoding equipment

Terminology used for these codes states that to a binary unit we assign a voltage or current level: mark (M) and to a binary zero we assign zero level: space(S): $1 \rightarrow M, 0 \rightarrow S$.

Differential coding is a technique used in BB coding as well as band pass modulation in order to obtain immunity to polarity inversion of the signal.

The block-schemes of the differential encoder and decoder are presented in Fig. 5.59

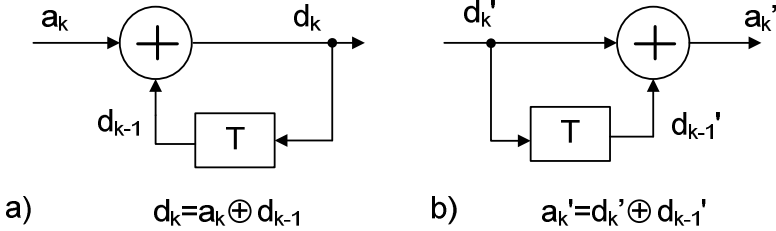


Fig. 5.59 Differential a) encoder and b) decoder;

where:

- a_k – the original binary data;
- d_k – the differentially encoded data;
- \oplus – modulo 2 adder
- d'_k – received differentially encoded data
- a'_k – decoded data

An example of differential coding is given in Table 5.17

Table 5.17 Example of differential coding.

a_k	1	0	0	1	1	0	1	0	0	1	
$a_k \oplus d_{k-1} = d_k$	1	0	0	0	1	0	0	1	1	1	0
d'_k correct polarity	1	0	0	0	1	0	0	1	1	1	0
$d'_k \oplus d_{k-1} = a'_k$	1	0	0	1	1	0	1	0	0	0	1
d'_k inverse polarity	0	1	1	1	0	1	1	0	0	0	1
$d'_k \oplus d_{k-1} = a'_k$	1	0	0	1	1	0	1	0	0	0	1

In differential decoding the signal is decoded pointing out the transitions, and not the absolute values of the signals, which allow high noise immunity. Another great advantage of differential coding is its insensitivity to the absolute polarity of the signal, meaning that an accurate decoding is possible although the polarity of a pair is accidentally converted.

Majority of BB codes can be classified in the following groups:

- a. non-return to zero(NRZ;NRZ-L,NRZ-M,NRZ-S)
- b. return to zero(RZ)
- c. biphasic(B, BL, BM, BS, Miller)
- d. pseudoternary(AMI, twined binary: dicode NRZ, dicode RZ)
- e. substitution(BNZS,HDBn,CMI,DMI)

In the followings the most important BB codes will be described.

- a. *NON-Return to Zero*: NRZ-L, NRZ-M,NRZ-S
 - *NRZ-L (Non Return to Zero Level)*: “1” is represented by a constant level during the bit duration T , and “0” by another constant level during T ; the code can be unipolar (0, A), (0, -A) or polar/antipodal signal (-A, +A);
 - *NRZ-M (Non Return to Zero Mark)* for a “1” there is a transition at the beginning of the bit interval: for “0” there is not level change;
 - *NRZ-S (Non Return to Zero Space)*: for a “0” there is a transition at the beginning of the bit interval; for “1” there is not level change;
 - NRZ-M and NRZ-S are differential versions of NRZ-L and have the advantages of any differential code over NRZ-L;
 - if the “0”, and “1” are equally probable in the signal and the signalling is polar (+A, -A), NRZ has no DC component; if the signal is unipolar (0,A), DC component is $A/2$.
 - *applications* of NRZ codes: NRZ-L is the basic code for digital logic (the terminals, for example); NRZ-M-is used in magnetic tape recording;
 - the lack of transitions for long strings of “0” or “1” means poor bit synchronization capacity and therefore, NRZ codes have limited applications in telecommunications.
- b. *Return to Zero (RZ)*: for a “1” a positive pulse of $T/2$ width is transmitted, thus the name of Return to Zero; for a “0”, zero level is transmitted during T , the bit duration, in a unipolar RZ code, or a negative pulse of $T/2$ width is transmitted for a polar RZ code
 - if “0”and “1” are equally probable, no DC component for a polar RZ and $A/4$ DC component for unipolar RZ;
 - synchronization capacity is good for polar RZ (there are two transitions per bit) but poor for unipolar RZ, when for long strings of “0” there are not transitions;
 - the pulse duration being $T/2$, and not T as for NRZ codes it means that the bandwidth of RZ codes is doubled compared to NRZ
- c. *Biphase codes*
 - *B-L (Biphase Level-Manchester)*: a ”0” is represented by 01 and a ”1” by 10; it means that each bit contains two states of $T/2$ duration. At each bit the signal phase changes with π , therefore the name biphase. This code is known also as Manchester code;

- *B-M (Biphase Mark)*: there is a transition at the beginning of each bit. For "1" the second transition occurs after $T/2$ and for "0", there is not the second transition until the beginning of the next bit.
- *B-S (Biphase Space)*: there is a transition at the beginning of each bit. For "0" the second transition occurs after $T/2$ and for "1" there is not the second transition until the beginning of the next bit.
- *Differential BL*: a transition takes place in the middle of each bit regardless it is 0 or 1. For "0" the first transition takes place at the beginning of the bit and the second occurs after $T/2$. For "1" the unique transition is at $T/2$.
- *Delay Modulation (Miller code)* is a variety of biphase code; for "1" a transition occurs at $T/2$. For "0", a transition takes place at the end of the bit if it is followed by a "0"; if a "1" comes next, there is no transition at the end of the bit. As shows in Fig 5.61, Miller code is obtained from BL (Manchester) by suppressing each second transition. For Miller code there exist at least one transition every 2 bits (corresponding to the most unfavorable sequence:101) and there is never more than one transition per bit, which means that it has a good bit synchronization capacity and very small DC component, making from it an excellent code for magnetic recordings [20].

Main features of biphase codes:

- B-M and B-S are differential versions of B-L
- all the three biphase codes have at least one transition in a bit duration (T), this providing a very good bit synchronization;
- all the three biphase codes have not DC component if the signalling is polar;
- *applications*: magnetic recording, optical and satellite communications.

d. Pseudo ternary line codes

These binary codes are using three levels: +A, -A and 0, which explains the name of pseudo ternary (in telecommunications they are called often bipolar codes).

- *AMI (Alternate Mark Inversion)* is the most important code of this group. A "0" is encoded with zero level, during the whole bit duration. A "1" is encoded with level +A or -A, alternatively, from where the name of alternate mark inversion. AMI can be encoded both in NRZ and RZ forms; however the most used form is RZ. AMI has no DC component; identifying its use in digital telephony with scrambling.
 - *scramblers* are pseudo noise sequences, generated with LFSR (see 5.8.5) used to ensure $p(0) \approx p(1)$ in any binary sequence, this improving the bit synchronization capacity of the signal and subsequently eliminating the main disadvantage;
 - redundancy in the waveform allows the possibility to monitor the quality of the line without knowing the nature of the traffic being transmitted. Since the "1"s alternate as polarity it means that any violation of this rule implies an error. In this way, the line code, without being properly an error control code (see cap.5), acts for error detection. The numbers of bipolar

violation are counted and if the frequency of their occurrence exceeds some threshold, an alarm is set. This error detection is done in the traffic supervising the line code

- AMI has been widely used in digital coaxial or pair cable systems (in early T1-digital multiplex of 1,544 Mbps and E1-European primary multiplex of 2,048Mbps) [4], [9];
- *Dicode (twinned binary)* [11], [49] are related to differential coding. If the data sequence to be encoded is $\{a_k\}$, a dicode is obtained encoding AMI the sequence $\{d_k\}$, where $d_k = a_{k-1} - a_k$.

Example: $\{a_k\} = \{ 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1 \}$ and $a_0 = 1$
 $\{d_k\} = \{ 0, 1, -1, 1, -1, 1, 0, 0, 0, -1, 0, 1, 0, 0, 0, -1 \}$

- Dicode can be represented in NRZ or RZ (Fig. 5.61)

e. *Substitution line codes*

AMI code, despite its great advantages: no DC component and error detection capacity has a great draw-back: without scrambler, bit synchronization capacity is very poor; long strings of “0”s, which is very frequent in telephony, means no bit timing recovery. In order to avoid this major disadvantages without scrambling there are two popular zero substitution codes: binary N-zero substitution (BNZS) and high density bipolar n(HDB n) codes.

- *Binary N-Zero Substitution Codes (BNZS)*, were proposed by V.I.Johannes in 1969 [25]. It replaces a string of N zeros in AMI format with a special N-bit waveform with at least one bipolar violation. Thus density of pulses is increased while the original data are obtained recognizing the bipolar violation and replacing them at receiver with N zeros.

Example: B3ZS, a code of three zero substitution, used in DS-3 signal interface in North America standard and in LD-4 coaxial transmission system in Canada [4].

- 0 0 0 → 000V, where V means bipolar violation
 → B0V, where B means a single pulse with bipolar alternation

In both substitution, the bipolar violation occurs in the last position of the three zeros replaced by the code, and this position is easily identified. The decision to substitute with 00V or B0V is made according to the number of bipolar pulses (B) transmitted since the last substitution:

- if odd → 0 0 V
- if even → B 0 V

In this way the encoded signal has an odd number of bipolar violation. Also, bipolar violation alternate in polarity, so the DC component is missing.

An even number of bipolar pulses between bipolar violations occurs only as a result of channel error.

Other BNZS used in telephony are: B6ZS [4], B8ZS [4].

- *HDBn (High Density Bipolar n)* was proposed by A.Croisier [11] in 1970. These codes limit the number of consecutive zeros to n, replacing the (n+1) consecutive zeros by a sequence with bipolar violation.

Example HDB-3:

- 0000 → 0 0 0 V if m is odd
- 0000 → B 0 0 V if m is even

where m represents the number of consecutive marks since the last substitution.

HDB3 is used in 2.048Mbps, 8.448Mbps and 34.368 Mbps multiplex within the European digital hierarchy [9].

In optical communication systems a symbol is represented by the intensity of the light given by the optical source, the laser diode. This is why a ternary code (as AMI), can not be used. A solution to avoid this difficulty was to replace the zero level of AMI with two-level waveforms.

- *Coded Mark Inversion (CMI)*, proposed by Takasaki [44] in 1976, uses alternate signalling (A, -A) for “1”s in NRZ format, and for “0” a sequence 01. CMI enhances the transition density and thus the timing capacity and based on the alternation of “1”s, has error detection capacity through the monitoring of polarity violation (as AMI). It is used in the 139.246 Mbps multiplex within the European digital hierarchy [9].
- *Differential Mode Inversion Code (DMI)*, proposed also by Takasaki [44] is similar to CMI for coding “1”s and for “0”s differ in such a way to avoid pulses larger than T: 0->01 or 10.

f. Multilevel Signalling

Many of the baseband codes presented so far are binary, using two-levels ($D = M$). There are applications with limited bandwidth and higher data rates requirements. Data rate can be enhanced increasing the number of signalling levels

(increasing the alphabet $m > 2$) while keeping the same signalling rate $\dot{M} = \frac{1}{T}$.

Data rate (D) of a multilevel system is (2.27):

$$D = \dot{M} \cdot \log_2 m = \frac{1}{T} \log_2 m$$

In this case the moment rate \dot{M} is known as symbol rate. An example of quaternary signalling: $m=4$ is achieved by 2 bits per signal interval (T): $00 \rightarrow -3A$, $01 \rightarrow -A$, $10 \rightarrow +A$ and $11 \rightarrow +3A$.

Error performance of line codes. Each application requires a minimum BER at the user. In transmission systems, if this value is obtained at lower SNR, the regenerative repeaters can be spaced farther, this reducing installation and maintenance costs [4].

A thorough analysis and demonstrations of BER for most line codes can be found in [49].

Further we present a short description of signal detection at receiver. Basics of signal detection are found in Appendix C. For binary symbols, detection means to take decision of which signal was transmitted, by comparing at the appropriate time (at $T/2$, T , according to the type of decision) the measurement with a threshold located halfway between the two levels (0 in polar case, $A/2$ for unipolar and 0, $+A/2$, $-A/2$ for bipolar signalling). The BER, after detections is a function of SNR.

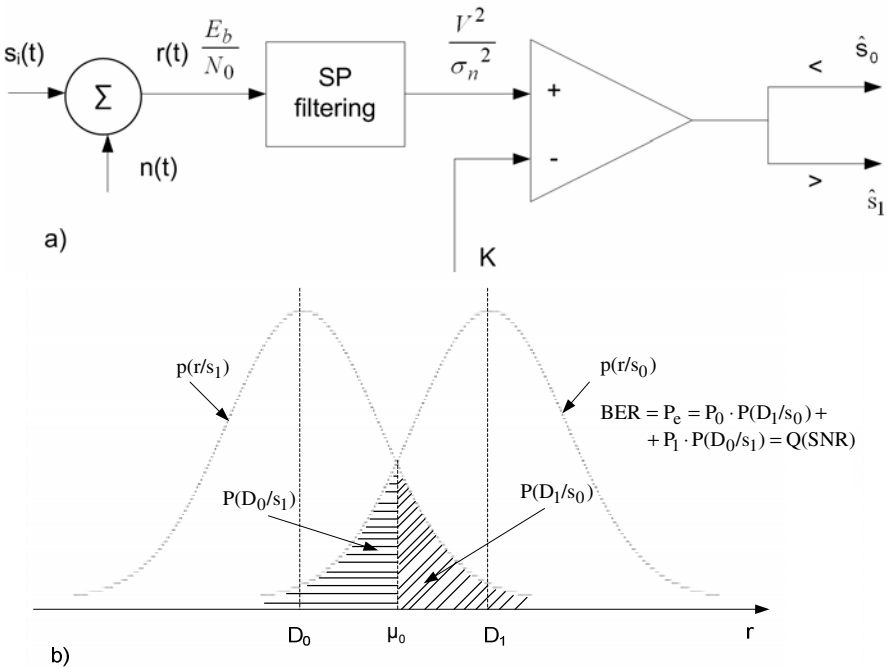


Fig. 5.60 Signal detection: a) block-scheme and b) illustration of BER

The significance of the notations in Fig. 5.60 is:

- $s_i(t) = \begin{cases} s_1(t) \\ s_0(t) \end{cases}$ - the binary symbols
- $n(t)$ = noise, assumed AWGN (Additive White Gaussian Noise)
- $r(t) = s_i(t) + n(t)$ - received signal
- E_b = energy per bit
- N_0 = spectral density power of AWGN
- V^2 = power of the signal at the decision block (comparator)
- σ_n^2 = power of the noise (R is assumed unitary)
- $p(r/s_i)$ = probability density function of r/s_i
- D_0, D_1 = decision \hat{s}_0 , respectively \hat{s}_1
- \hat{s}_0, \hat{s}_1 = estimated binary signals after decision
- P_e = bit error rate (BER)
- Q = erfc (see Appendix C)

The optimal receiver for binary signals is using a continuous observation and can be implemented using a correlated or a matched filter [27], [45] and (Appendix C).

Polar signalling (A, -A) is optimal because it maximizes the SNR; in this case the two signals are identical, but opposite (antipodal signalling). For this reason it is used often as basis for comparisons. The line codes with polar signalling are NRZ and biphasic.

Unipolar signalling codes use the same detector as for polar signal detection. The only difference is that the decision threshold is moved from 0 to $A/2$. To keep the same BER as in polar case, the power of the signal need to be doubled, meaning that unipolar signalling is 3dB less with respect to polar performance (see Appendix C).

Bipolar signalling (AMI and derived: decode, BNZS, HDBn) from error performance point of view is identical to unipolar code. During any interval the receiver must decide between 0 and $\pm A/2$. A slightly higher BER than in unipolar signalling occurs because both positive and negative noise may cause an erroneous threshold, crossing when a zero-level signal is transmitted. A graphical representation of BER for these three types of signalling is given in Fig. 4.25, pag. 198 from [4].

Multilevel signalling requires the same bandwidth as binary systems ($M = \frac{1}{T}$ is the same), but uses $m > 2$ levels instead of two. The penalty is the increase of signal power in order to keep the same BER (see quaternary signalling Fig.5.61, which uses $\pm 3A$ levels in addition). A graphical representation is given in Fig. 4.26, pag.201 from [4].

Comparative analysis of line codes

Comparison of line codes is made having in mind the aspects mentioned at the beginning of this chapter:

- *signal spectrum*
 - no DC component (advantage): biphasic codes, AMI, dicode, HDB-n, BNZS, CMI, DMI
 - with DC component (drawback): NRZ, RZ
 - bandwidth in increasing order: multilevel coding, Miller, NRZ, biphasic, RZ, dicode, CMI, DMI, BNZS
- *bit synchronization capacity*:
 - good: biphasic codes, BNZS, CMI, DMI
 - poor: NRZ, RZ, AMI
- *code efficiency* (η), defined by (5.301);
 - NRZ: 1 bit is encoded into 1 binary symbol

$$\eta_{\text{NRZ}} = \frac{\log_2 2}{\log_2 2} = 1 (100\%)$$
 - Manchester: 1 bit is encoded into 2 binary symbols

$$\eta_{\text{M}} = \frac{\log_2 2}{2 \cdot \log_2 2} = \frac{1}{2} = 0,5 (50\%)$$
 - AMI: 1 bit is encoded into 1 ternary symbol

$$\eta_{\text{AMI}} = \frac{\log_2 2}{\log_2 3} = \frac{1}{1,59} \approx 0,63 (63\%)$$
- *error performance*, as presented before, is the best for polar (antipodal) codes, than follow unipolar codes and bipolar codes.
- *error detection capacity* is possible for AMI, HDBn, CMI, and DMI, based on the violation of encoding laws, without being properly error control codes.
- *immunity to polarity inversion* is given by all differential codes.

Example 5.38

Encode in all presented base-band codes the sequence:

101010100001100001

The encoding is presented in Fig. 5.61.

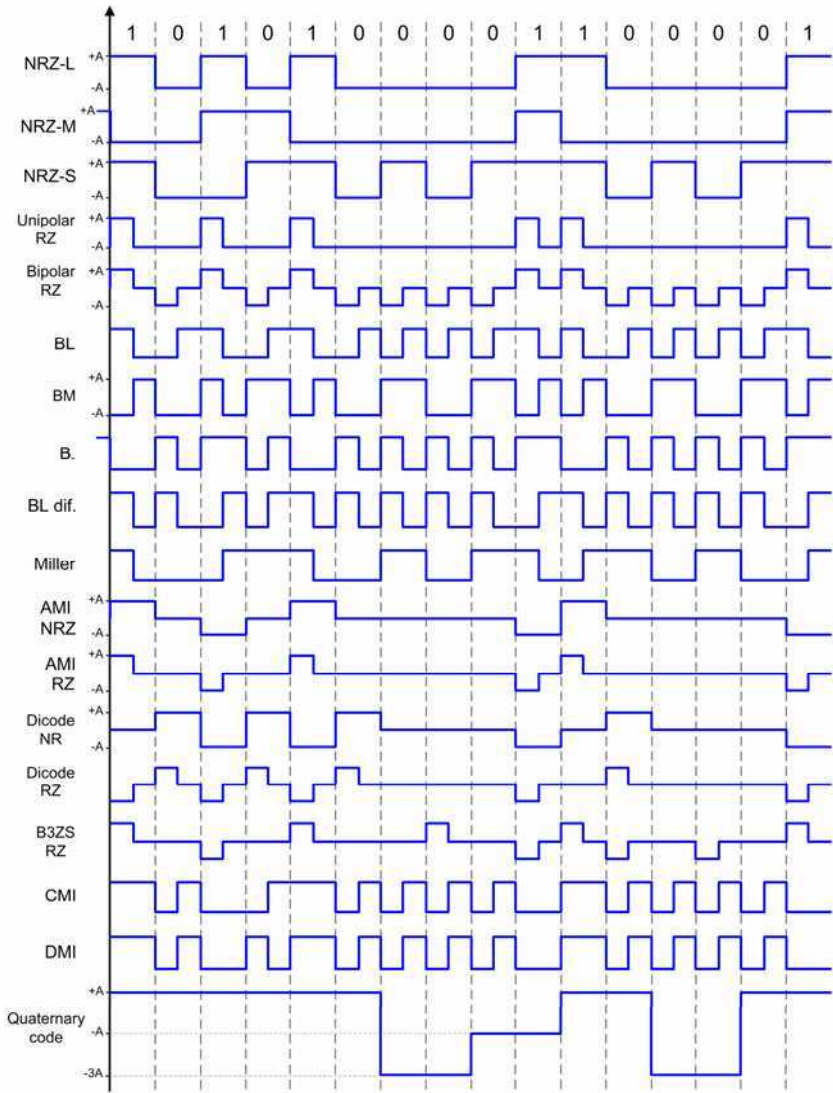


Fig. 5.61 Examples of Base Band encoding

References

- [1] Aaron, M.R.: PCM transmission in the exchange plant. *Bell System Technical Journal* 41, 99–141 (1962)
- [2] Angheloiu, I.: *Teoria codurilor*. Editura Militara, Bucuresti (1972)
- [3] Benice, R., Frey, A.: An analysis of retransmission systems. *IEEE Transactions on Comm. Tech.*, 135–145 (1964)
- [4] Bellamy, J.: *Digital Telephony*, 2nd edn. John Wiley & Sons, New York (1991)
- [5] Berrou, C., Glavieux, A.: Near Shannon limit error-correcting coding and decoding turbo-codes. In: *Proc. ICC 1993, Geneva*, pp. 1064–1070 (1993)
- [6] Blahut, R.: *Algebraic Methods for Signal Processing and Communication Coding*. Springer, Heidelberg (1992)
- [7] Bose, R., Chandhuri, D.R.: On a class of error correcting binary group codes. *Inf. Control* 3, 68–70 (1960)
- [8] CCITT yellow book, *Digital Networks-Transmission systems and multiplexing equipment vol. III.3*, Geneva, ITU (1981)
- [9] Chen, C.: High-speed decoding of BCH codes. *IEEE Trans. on Inf. Theory* IT 2, 254–256 (1981)
- [10] Chien, R.: Cyclic decoding procedure for the BCH codes. *IEEE Trans. Inf. Theory* IT 10, 357–363 (1964)
- [11] Croisier, A.: Introduction to pseudo ternary transmission codes. *IBM Journal of Research and Development*, 354–367 (1970)
- [12] Cullmann, G.: *Codage et transmission de l'information*. Editions Eyrolles, Paris (1972)
- [13] Erfanian, J., Pasupathy, S., Gulak, G.: Reduced complexity symbol detectors with parallel structure for ISI channels. *IEEE Transactions on Communications* 42(234), 1661–1671 (1994)
- [14] Evans, M.: Nelson Matrix Can Pin Down 2 Errors per Word. *Electronics* 55(11), 158–162 (1982)
- [15] Fontolliet, P.: *Systemes de telecommunications*. Editions Georgi, Lausanne (1983)
- [16] Forney, G.: On decoding binary BCH codes. *IEEE Trans. on Inf. Theory* IT 11, 580–585 (1965)
- [17] Forney Jr., G.D.: *Concatenated codes*. MIT Press, Cambridge Mass (1966)
- [18] Hamming, R.: *Coding and information theory*. Prentice-Hall, Englewood Cliffs (1980)
- [19] Haykin, S.: *Communication Systems*, 4th edn. John Wiley & Sons, Chichester (2001)
- [20] Hecht, M., Guida, A.: Delay modulation. *Proceedings IEEE* 57(7), 1314–1316 (1969)
- [21] Hanzo, L., Liew, T.H., Yeap, B.L.: Turbo Coding. In: *Turbo Equalisation and Space-Time Coding for Transmission over Fading Channels*, Wiley-IEEE Press, Chichester (2002)
- [22] Hocquenghem, A.: Codes correcteurs d'erreurs. *Cliffres* 2, 147–156 (1959)
- [23] Hsiao, M.: A class of optimal minimum odd-weight-column SEC-DED codes. *IBM Journal of Research and Development* 14(4), 395–401 (1970)
- [24] Ionescu, D.: *Codificare si coduri*. Editura Tehnica, Bucuresti (1981)
- [25] Johannes, V.I., Kaim, A.G., Walzman, T.: Bipolar pulse transmission with zero extraction. *IEEE Transactions in Communications Techniques* 17, 303 (1969)
- [26] Johnston, B., Johnston, W.: LD-4. A Digital Pioneer in Action. *Telesis* 5(3), 66–72 (1977)

- [27] Kay, S.M.: Fundamentals of Statistical Signal Processing. In: Kay, S.M. (ed.) Detection Theory, vol. 1, Prentice Hall, Englewood Cliffs (1998)
- [28] Lin, S., Costello, D.: Error control coding. Prentice-Hall, Englewood Cliffs (1983)
- [29] Lidl, R., Niederreiter, H., Cohn, P.M.: Finite Fields. Cambridge University Press, Cambridge (1997)
- [30] Massey, J.: Shift register synthesis and BCH decoding. IEEE Trans. on Inf. Theory IT 17, 464–466 (1971)
- [31] Mateescu, A.: Manualul inginerului electronist. Transmisuni de date, Editura Tehnica, Bucuresti (1983)
- [32] Odenwalder, J.P.: Error control coding handbook. Linkabit Corporation, San Diego (1976)
- [33] Omura, J.K.: On the Viterbi decoding algorithm. IEEE Trans. on Inf. Theory IT 15, 177–179 (1969)
- [34] Patel, A., Hong, S.: Optimal rectangular code for high density magnetic tapes. IBM J. Res. Dev. 18, 579–588 (1974)
- [35] Patel, A.M.: Error recovery scheme for the IBM 3850 mass storage system. IBM J. Res. Dev. 24, 32–44 (1980)
- [36] Peterson, W., Weldon, W.: Error correcting codes. MIT Press, Cambridge (1972)
- [37] Price, W., Barber, D.: Teleinformatica - Retele de calculatoare si protocoalele lor. Editura Tehnica, Bucuresti (1983)
- [38] Qualcomm, Inc., Q1401, Viterbi decoder device information (1987)
- [39] Radu, M., et al.: Telefonie numerica. Editura Militara, Bucuresti (1988)
- [40] Reed, I., Solomon, G.: Polynomial codes over certain finite fields. J. Soc. Ind. Appl. Math. 8, 300–304 (1960)
- [41] Shannon, C.: A Mathematical Theory Of Communication. Bell System Technical Journal 27, 379–423 (1948); Reprinted in Shannon Collected Papers, IEEE Press (1993)
- [42] Sklar, B.: Digital communications. Prentice-Hall, Englewood Cliffs (1988)
- [43] Spataru, A.: Fondements de la theorie de la transmission de linformation. Presses Polytechniques Romandes, Lausanne (1987)
- [44] Takasaki, Y., et al.: Optical pulse formats for fiber optic digital communications. IEEE Transactions on Communications 24, 404–413 (1976)
- [45] Van Trees, H.L.: Detection, Estimation and Modulation Theory, Part.I. John Wiley & Sons, Chichester (1968)
- [46] Vucetic, B., Yuan, J.: Turbo codes. Kluwer Academic Publishers Group, Boston (2001) (2nd printing)
- [47] Wade, G.: Signal coding and processing. Cambridge University Press, Cambridge (1994)
- [48] Wade, G.: Coding Techniques. Palgrave (2000)
- [49] Xiong, F.: Digital Modulation Techniques. Artech House (2000)
- [50] Gallager, R.G.: Information Theory and Reliable Communication. John Willey & Sons, New York (1968)
- [51] Slepian, D.: Key Papers in the Development of Information Theory. IEEE Press, Los Alamitos (1974)
- [52] Shparlinski, I.: Finite Fields: Theory and Computation. Kluwer Academic Publishers Group, Boston (1999)
- [53] Lidl, R., Niederreiter, H.: Finite Fields, Encyclopedia of Mathematics and Its Applications, vol. 20. Addison-Wesley Publ. Co., Reading (1983)

- [54] McEliece, R.J.: *The Theory of Information and Coding*, 2nd edn. Cambridge University Press, Cambridge (2002)
- [55] Heegard, C., Wicker, S.B.: *Turbo Coding*. Kluwer Academic Publishers Group, Boston (1999)
- [56] Lee, C.L.H.: *Convolutional Coding – Fundamentals and Applications*. Artech House, Boston (1997)
- [57] Wozencraft, J.M., Jacobs, I.M.: *Principles of Communication Engineering*. Waveland Press, Prospect Heights (1990)
- [58] Peterson, W.W., Weldon Jr., E.J.: *Error-correcting Codes*, 2nd edn. MIT Press, Cambridge (1972)
- [59] Ramsey, J.: Realization of optimum interleavers. *IEEE Transactions on Information Theory* 16(3), 338–345 (1970)
- [60] Forney Jr., G.: Burst-Correcting Codes for the Classic Bursty Channel. *IEEE Transactions on Communication Technology* 19(5), 772–781 (1971)

Appendix A: Algebra Elements

A1 Composition Laws

A1.1 Compositions Law Elements

Let us consider M a non empty set. An application φ defined on the Cartesian product $M \times M$ with values in M :

$$M \times M \rightarrow M, (x, y) \rightarrow \varphi(x, y) \quad (\text{A.1})$$

is called *composition law* on M ; it defines the effective law by which to any ordered pair (x, y) of M elements is associated a unique element $\varphi(x, y)$, which belongs to the set M as well.

The mathematical operation in such a law can be noted in different ways: $+$, $-$, $*$, 0 etc. We underline the fact that the operations may have no link with the addition or the multiplication of numbers.

A1.2 Stable Part

Let us consider M a set for which a composition law is defined and H a subset of M . The set H is a *stable part* of M related to the composition law, or is closed towards that law if:

$$\forall x, y \in H : \varphi(x, y) \in H$$

where φ is the composition law.

Example

- The set \mathbf{Z} of integer numbers is a stable part of the real numbers set \mathbf{R} towards addition and multiplication.
- The natural numbers set \mathbf{N} is not a stable part of the real numbers set \mathbf{R} towards subtraction.

A1.3 Properties

The notion of composition law presents a high degree of generality by the fact that the elements nature upon which we act and the effective way in which we act are ignored.

The study of composition laws based only on their definition has poor results. The idea of studying composition laws with certain properties proved to be useful, and these properties will be presented further on. For the future, we will assume the fulfilment of the law:

$$M \times M \rightarrow M, (x, y) \rightarrow x * y$$

Associativity: the law is associative if for $\forall x, y, z \in M$:

$$(x * y) * z = x * (y * z) \quad (\text{A.2})$$

If the law is *additive*, we have:

$$(x + y) + z = x + (y + z)$$

and if it is *multiplicative*, we have:

$$(xy)z = x(yz)$$

Commutativity: the law is commutative if for $\forall x, y, z \in M$:

$$x * y = y * x \quad (\text{A.3})$$

Neutral element: the element $e \in M$ is called neutral element if:

$$e * x = x * e = x, \forall x \in M \quad (\text{A.4})$$

It can be demonstrated that if it exists, then it is unique.

For real numbers, the neutral element is 0 in addition and 1 in multiplication and we have:

$$x + 0 = 0 + x; \quad x \cdot 1 = 1 \cdot x = x$$

Symmetrical element: an element $x \in M$ has a symmetrical element referred to the composition law $*$, if there is an $x' \in M$ such that:

$$x' * x = x * x' = e \quad (\text{A.5})$$

where e is the neutral element.

The element x' is the symmetrical (with respect to x) of x .

From the operation table (a table with n rows and n columns for a set M with n elements), we can easily deduce whether the law is commutative, whether it has neutral element or whether it has symmetrical element. Thus:

- if the table is symmetrical to the main diagonal, the law is commutative
- if the line of an element is identical with the title line, the element is neutral one
- if the line of an element contains the neutral element, the symmetrical of that element is to be found on the title line on that column where the neutral element belongs.

Example

Be the operation table:

*	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

- neutral element is 1
- operation is commutative
- symmetrical of 2 is 3 and so on.

A2 Modular Arithmetic

In technical applications, the necessity of grouping integer numbers in sets according to remainders obtained by division to a natural number n , frequently occurs.

Thus, it is known that for any $a \in \mathbf{Z}$, there is a $q, r \in \mathbf{Z}$, uniquely determined, so that:

$$a = n \cdot q + r, \quad r = 0, 1, \dots, n-1 \quad (\text{A.6})$$

The set of numbers divisible to n , contains the numbers which have the remainders $1, \dots, \text{the remainder } n-1$, and they are noted with $\hat{0}, \hat{1}, \dots, \hat{n}-1$; they are giving the congruence *modulo n classes*, denoted with \mathbf{Z}_n .

The addition and multiplication are usually noted with \oplus and \otimes . The addition and multiplication are done as in regular arithmetic.

Example

For \mathbf{Z}_5 , we have:

\oplus	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

\otimes	0	1	2	3	4
0	0	1	2	3	4
1	1	1	2	3	4
2	2	2	4	1	3
3	3	3	1	4	2
4	4	4	3	2	1

For subtraction, the additive inverse is added: $2 - 4 = 2 + 1 = 3$, because the inverse of 4 is 1. It is similar for division: $2/3 = 2 \cdot 1/3 = 2 \cdot 2 = 4$, because the multiplicative inverse of 3 is 2.

Remark

These procedures will be used for more general sets than the integer numbers.

A3 Algebraic Structures

By *algebraic structure*, we understand a non zero set M characterized by one or more composition laws and which satisfy several properties, from the above mentioned ones, known as structure axioms. For the problems that we are interested in, we will use two structures, one with a composition law called *group* and the other one with two composition laws, called *field*. Some other related structures will be mentioned too.

A3.1 Group

A joint $(G,*)$ formed by a non empty set G and with a composition law on G :

$$G \times G \rightarrow G, (x, y) \rightarrow x * y \text{ with } x, y \in G$$

is called *group* if the following axioms are met:

- associativity: $(x * y) * z = x * (y * z), \forall x, y, z \in G$
- neutral element:

$$\exists e \in G, e * x = x * e = x \quad \forall x \in G \tag{A.7}$$

- symmetrical element; when the commutativity axiom is valid as well: $x * y = y * x, \forall x, y \in G$ the *group* is called *commutative* or *Abelian group*.

If G has a finite number of elements, the group is called *finite group of order m*, where m represents the elements number.

Remarks

1. In a group, we have the simplification rules to the right and to the left:

$$a * b = a * c \Rightarrow b = c \tag{A.8}$$

$$b * a = c * a \Rightarrow b = c \tag{A.9}$$

2. If in a group $(G,*)$, there is a set $H \subset G$, so that $(H,*)$ should at its turn form a group, this is called *subgroup of G*, having the same neutral element and inverse as G .
3. If the structure contains only the associative axiom and the neutral element, it is called *monoid*.

Example

The integer numbers form a group related to addition, but not related to multiplication, because the inverse of integer k is $1/k$, which is not an integer for $k \neq 1$.

The congruence modulo any n classes, are Abelian groups related to addition, and the ones related to multiplication are not Abelian groups unless the module n is prime, as we see in table for \mathbf{Z}_5 . When the module is not prime, the neutral element is 1 as well, but there are elements that do not have symmetrical number, for example element 2 in \mathbf{Z}_4 :

\otimes	1	2	3
1	1	2	3
2	2	0	2
3	3	2	1

A3.2 Field

A non empty set A with two composition laws (conventionally named addition and multiplication), and symbolised with $+$ and \bullet , is called *field* if:

- $(A,+)$ is an Abelian group
- (A_1,\bullet) is an Abelian group, $A_1 = A/\{0\}$ where “0” is the neutral element of $(A,+)$
- Distributivity of multiplication related to addition: $x(y+z) = xy + xz$

Remarks:

- if (A_1,\bullet) is a group, without being Abelian, the structure is called *body*; so the *field* is a *commutative body*. If (A_1,\bullet) is monoid only, then the structure is *ring*.
- the congruence modulo p classes, with p prime, form a ring. Rings may contain divisors of 0, so non zero elements with zero product. In the multiplication example \mathbf{Z}_4 we have $2*2 = 0$, so 2 is a divisor of 0. These divisors of 0 do not appear in bodies.

A3.3 Galois Field

A field can have a finite number m of elements in A . In this case, the field is called *m degree finite field*. The minimum number of elements is 2, namely the neutral elements of the two operations, so with the additive and multiplicative notations: 0 and 1. In this case, the second group contains a single element, the unit element 1. The operation tables for both elements are in \mathbf{Z}_2 :

\oplus	0	1
0	0	1
1	1	0

\otimes	0	1
0	0	0
1	0	1

This is the binary field, noted with $\mathbf{GF}(2)$, a very used one in digital processing. If p is a prime number, \mathbf{Z}_p is a field, because $\{1, 2, \dots, p-1\}$ form a group with modulo p multiplication.

So the set $\{1, 2, \dots, p-1\}$ forms a field related to modulo p addition and multiplication.

This field is called *prime field* and is noted by $\mathbf{GF}(p)$.

There is a generalisation which says that, for each positive integer m , we should extend the previous field into a field with p^m elements, called the *extension of the field* $\mathbf{GF}(p)$, noted by $\mathbf{GF}(p^m)$.

Finite fields are also called *Galois fields*, which justifies the initials of the notation \mathbf{GF} (Galois Field).

A great part of the algebraic coding theory is built around finite fields. We will examine some of the basic structures of these fields, their arithmetic, as well as the field construction and extension, starting from prime fields.

A.3.3.1 Field Characteristic

We consider the finite field with q elements $\mathbf{GF}(q)$, where q is a natural number.

If 1 is the neutral element for addition, be the summations:

$$\sum_{i=1}^1 1 = 1, \sum_{i=1}^2 1 = 1 + 1 = 2, \dots, \sum_{i=1}^k 1 = 1 + 1 + \dots + 1$$

\downarrow
 $k \text{ terms}$

As the field is closed with respect to addition, these summations must be elements of the field.

The field having a finite number of elements, these summations cannot all be distinct, so they must repeat somewhere; there are two integers m and n ($m < n$), so that

$$\sum_{i=1}^m 1 = \sum_{i=1}^n 1 \Rightarrow \sum_{i=1}^{n-m} 1 = 0$$

There is the smallest integer λ so that $\sum_{i=1}^{\lambda} 1 = 0$. This integer is called *the characteristic of the field* $\mathbf{GF}(q)$.

The characteristic of the binary field $\mathbf{GF}(2)$ is 2 , because the smallest λ for which

$$\sum_{i=1}^{\lambda} 1 = 0 \text{ is } 2, \text{ meaning } 1 + 1 = 0$$

The characteristic of the prime field $\mathbf{GF}(p)$ is p . It results that:

- the characteristic of a finite field is a prime number
- for $n, m < \lambda$, $\sum_{i=1}^n 1 \neq \sum_{i=1}^m 1$

- the summations: $1, \sum_{i=1}^2 1, \sum_{i=1}^3 1, \dots, \sum_{i=1}^{\lambda-1} 1, \sum_{i=1}^{\lambda} 1 = 0$: are λ distinct elements in $\mathbf{GF}(q)$, which form a field with λ elements $\mathbf{GF}(\lambda)$, called *subfield of $\mathbf{GF}(q)$* . Subsequently, any finite field $\mathbf{GF}(q)$ of characteristic λ contains a subfield with λ elements and it can be shown that if $q \neq \lambda$ then q is an exponent of λ .

A.3.3.2 Order of an Element

We proceed in a similar manner for multiplication: if a is a non zero element in $\mathbf{GF}(q)$, the smallest positive integer, n , so that $a^n = 1$ gives the *order of the element a* .

This means that $a, a^2, \dots, a^n = 1$ are all distinct, so they form a multiplicative group in $\mathbf{GF}(q)$.

A group is called *cyclic group* if it contains an element whose successive exponents should give all the elements of the group. If in the multiplicative group, there are $q-1$ elements, we have $a^{q-1} = 1$ for any element, so the order n of the group divides $q-1$.

In a finite field $\mathbf{GF}(q)$ an element a is called *primitive element* if its order is $q-1$. The exponents of such an element generate all the non zero elements of $\mathbf{GF}(q)$. Any finite field has a primitive element.

Example

Let us consider the field $\mathbf{GF}(5)$, we have:

$$2^1 = 2, 2^2 = 4, 2^3 = 3, 2^4 = 1 \text{ so } 2 \text{ is primitive}$$

$$3^1 = 3, 3^2 = 4, 3^3 = 2, 3^4 = 1 \text{ so } 3 \text{ is primitive}$$

$$4^1 = 4, 4^2 = 1, \text{ so } 4 \text{ is not primitive.}$$

A4 Arithmetics of Binary Fields

We may build a power of p . We will use only binary codes in $\mathbf{GF}(2)$ or in the extension $\mathbf{GF}(2^m)$. Solving equations and equation systems in \mathbf{Z}_2 is not a problem, as $1+1=0$, so $1=-1$.

The calculations with polynomials in $\mathbf{GF}(2)$ are simple too, as the coefficients are only 0 and 1.

The first degree polynomials are X and $X+1$, the second degree ones are $X^2, 1+X^2, X+X^2, 1+X+X^2$. Generally, there are 2^n polynomials of degree n , the general form of the n degree polynomial being:

$f_0 + f_1X + \dots + f_{n-1}X^{n-1} + f_nX^n$, meaning n coefficients, so that:

$$C_n^n + C_n^{n-1} + \dots + C_n^0 = 2^n \quad (\text{A.10})$$

We notice that a polynomial is divisible by $X+1$ only if it has an even number of non zero terms. An m degree polynomial is *irreducible polynomial* on $\mathbf{GF}(2)$ only if it does not have any divisor with a smaller degree than m , but bigger than zero. From the four second degree polynomials, only the last one is irreducible; the others being divided by X or $X+1$. The polynomial X^3+X+1 is irreducible, as it does not have roots 0 or 1, it cannot have a second degree divisor; $1+X^2+X^3$ is also irreducible.

We present further 4th and 5th degree irreducible polynomials.

The polynomial X^4+X+1 is not divided by X or $X+1$, so it does not have first degree factors. It is obvious that it is not divided by X^2 either. If it should be divided by X^2+1 , it should be zero for $X^2=1$, which, by replacement, leads to $1+X+1=X \neq 0$; it cannot be divided by X^2+X either, as this one is $X(X+1)$. Finally, when we divide it by X^2+X+1 we find the remainder 1. There is no need for us to look for 3rd degree divisors, because then it should also have first degree divisors. So the polynomial X^4+X+1 is irreducible.

There is a *theorem* stating that *any irreducible polynomial on $\mathbf{GF}(2)$ of degree m divides $X^{2^m-1}+1$.*

We can easily check whether the polynomial X^3+X+1 divides $X^{2^3-1}+1=X^7+1$; as $X^3+X=1$ we have $X^6=X^2+1$ and $X^7=X^3+X=X+1+X=1$, so $X^7+1=0$

An irreducible polynomial $p(X)$ of degree m is primitive, if the smallest positive integer n for which $p(X)$ divides X^n+1 is 2^m-1 . In other words, $p(X)$ must be the simultaneous solution of the binomial equations $X^{2^m-1}+1=0$ and $X^n+1=0$, with $n \leq 2^m-1$. This does not occur except if n is a proper divisor of 2^m-1 , as we shall show further on. If 2^m-1 is *prime*, it does not have own divisors (except 2^m-1 and 1), so *any irreducible polynomial is primitive as well.*

Thus we may see that X^4+X+1 divides $X^{15}+1$, but it does not divide any polynomial X^n+1 , with $1 \leq n \leq 15$, so it is primitive. The irreducible polynomial $X^4+X^3+X^2+X+1$ is not primitive because it divides X^5+1 . But if $m=5$, we have $2^5-1=31$, which is prime number, so all irreducible polynomials of 5th degree are primitive as well.

For a certain m there are several primitive polynomials. Sometimes (for coding), the tables mention only one that has the smallest number of terms.

We will demonstrate the affirmation that the binomial equations $X^m + 1 = 0$ and $X^n + 1 = 0$ with $m < n$, do not have common roots unless m divides n . In fact, it is known that the m degree roots, n respectively, of the unit are:

$$X = \cos \frac{2k\pi}{m} + i \cdot \sin \frac{2k\pi}{m}, \quad k = \overline{0, m-1} \tag{A.11}$$

$$X_1 = \cos \frac{2k_1\pi}{m} + i \cdot \sin \frac{2k_1\pi}{m}, \quad k_1 = \overline{0, m-1} \tag{A.12}$$

In order to have a common root, besides $X = 1$, we should have:

$$\frac{2k\pi}{m} = \frac{2k_1\pi}{m}, \rightarrow k = \frac{k_1 \cdot m}{n} \tag{A.13}$$

But $k \in \mathbf{Z}$, which is possible only if m and n have a common divisor noted d . The common roots are the roots of the binomial equation $X^d + 1 = 0$, the other ones are distinct, d being the biggest common divisor of m and n .

In order to find the irreducible polynomials related to a polynomial, these ones must be irreducible in the modulo-two arithmetic, so they should not have a divisor smaller than them.

The first degree irreducible polynomials are X and $X + 1$. In order that a polynomial be not divisible to X , the free term must be 1, and in order not to be divisible to $X + 1$, it should have an odd number of terms.

For the 2nd degree polynomials, the only irreducible one is $X^2 + X + 1$.

For the 3rd, 4th and 5th degree polynomials, we shall take into account the previous notes and we shall look for those polynomials which are not divided by $X^2 + X + 1$.

The remainder of the division is obtained replacing in the polynomial: $X^2 + X + 1$, $X^3 = 1$, $X^4 = X$, etc.

For the 3rd degree irreducible polynomials, which should divide: $X^3 + \alpha X^2 + \beta X + 1$, one of the coefficients must be zero, otherwise the total number of terms would be even. Similarly, taking into account the previous notes, the remainder of the polynomial divided by $X^2 + X + 1$ is: $(\alpha + \beta)X + \alpha$.

We will have the following table:

α	β	$(\alpha + \beta)X + \alpha$	Polynomial	Irreducible	Primitive
1	0	$X + 1 \neq 0$	$X^3 + X^2 + 1$	YES	YES
0	1	$X \neq 0$	$X^3 + X + 1$	YES	YES

For each of the two cases, as the remainder is non zero, the polynomial is irreducible. It is obtained by replacing in the general form the corresponding values of the coefficients α and β .

For the 4th degree irreducible polynomials, we note the polynomial by: $X^4 + \alpha X^3 + \beta X^2 + \gamma X + 1$.

In order to have the total number of terms odd, all coefficients, or only one of them, must equal 1. As the remainder of the division by $X^2 + X + 1$ is: $X(1 + \beta + \gamma) + \alpha + \beta + 1$, we have the following table:

α	β	γ	$X(1+\beta+\gamma)+\alpha+\beta+1$	Polynomial	Irreducible	Primitive
1	1	1	$X+1 \neq 0$	$X^4+X^3+X^2+X+1$	YES	NO
1	0	0	$X \neq 0$	X^4+X^3+1	YES	YES
0	1	0	$X \cdot 2 + 2 = 0$	$X^4+X^2+1 = (X^2+X+1)^2$	NO	NO
0	0	1	$X \neq 0$	X^4+X+1	YES	YES

The first and the third are not primitive, as they divide $X^5 - 1$ and $X^3 - 1$, respectively, with a lower degree than $X^{15} - 1$. Indeed, 3 and 5 are divisors of 15.

Further on, we will present an important property of the polynomials on $\mathbf{GF}(2)$:

$$[f(X)]^2 = f(X^2) \tag{A.14}$$

Proof

Be $f(X) = f_0 + f_1X + \dots + f_nX^n$, we will have:

$$f^2(X) = f_0^2 + f_1^2X^2 + \dots + f_n^2X^{2n} = f_0 + f_1X^2 + f_2(X^2)^2 + \dots + f_n(X^2)^n = f(X^2),$$

as $1+1=0$ and $f_i^2 = f_i$.

A5 Construction of Galois Fields $\mathbf{GF}(2^m)$

We want to set up a Galois field with 2^m elements ($m > 1$) in the binary field $\mathbf{GF}(2)$, starting from its elements 0, 1 with the help of a new symbol α , as follows:

$$\begin{aligned} \mathbf{0} \cdot \mathbf{0} = \mathbf{0}, \mathbf{0} \cdot \mathbf{1} = \mathbf{1} \cdot \mathbf{0} = \mathbf{0}, \mathbf{1} \cdot \mathbf{1} = \mathbf{1}, \mathbf{0} \cdot \alpha = \mathbf{0}, \mathbf{1} \cdot \alpha = \alpha \cdot \mathbf{1} = \alpha \\ \alpha^2 = \alpha \cdot \alpha, \dots, \alpha^j = \underbrace{\alpha \cdot \alpha \cdot \dots \cdot \alpha}_j \text{ times}, \alpha^i \cdot \alpha^j = \alpha^j \cdot \alpha^i = \alpha^{i+j} \end{aligned} \tag{A.15}$$

We will consider the set:

$$F = \{ \mathbf{0}, \mathbf{1}, \alpha, \dots, \alpha^j, \dots \} \tag{A.16}$$

Be the set $F = \{ \mathbf{0}, \mathbf{1}, \alpha, \dots, \alpha^j, \dots \}$ to contain 2^m elements and to be closed related to the above multiplication. Be $p(X)$ a primitive polynomial of degree m with coefficients in $\mathbf{GF}(2)$. We suppose that $p(\alpha) = 0$. As $p(X)$ divides: $X^{2^m-1} + 1$

$$\exists q(X): X^{2^m-1} + 1 = q(X)p(X) \quad (\text{A.17})$$

Replacing X by α in the relationship (A.17), we will obtain:

$$\alpha^{2^m-1} + 1 = q(\alpha) \cdot p(\alpha) = 0 \quad (\text{A.18})$$

so:

$$\alpha^{2^m-1} = 1 \quad (\text{A.19})$$

With the condition $p(\alpha) = 0$, the set F becomes finite and will contain the elements:

$$F^* = \left\{ \mathbf{0} \quad \mathbf{1} \quad \alpha \quad \alpha^2 \quad \dots \quad \alpha^{2^m-2} \right\} \quad (\text{A.20})$$

The non zero elements of F^* are closed to the previously defined multiplication, which is easily demonstrated as follows:

Be $\alpha^i, \alpha^j \in F^*$, $i, j < 2^m - 1$; if $i + j > 2^m - 1$ we have:

$$i + j = (2^m - 1) + r, \quad \alpha^i \cdot \alpha^j = \alpha^{i+j} = \alpha^{(2^m-1) + r} = \alpha^r, \quad r < 2^m - 1 \quad (\text{A.21})$$

The set F is thus closed with respect to multiplication. In order that this set to be field, it needs to fulfil the field axioms.

From (A.15) and (A.16), we can see that the multiplication is commutative and associative having the neutral element 1. The inverse of α^i is $\alpha^{(2^m-1)-i}$.

The elements $\mathbf{1}, \alpha, \alpha^2, \alpha^{2^m-2}$ being distinct, they determine a group with the operation \bullet .

We will further define the addition operation "+" on F^* , so that the elements should form a group with the operation "+".

In order to facilitate the definition, we will first express the elements of the set F^* with the help of polynomials, checking the group axioms.

Be $p(X)$ a primitive polynomial of degree m and $0 \leq i \leq 2^m - 1$, where i is the degree of a polynomial X^i . We divide the polynomial by $p(X)$:

$$X^i = q_i(X)p(X) + a_i(X), \quad \deg a_i \leq m-1 \quad (\text{A.22})$$

The form of the remainder $a_i(X)$ is:

$$a_i(X) = a_{i0} + a_{i1}X + \dots + a_{i(m-1)}X^{m-1} \quad (\text{A.23})$$

Since X^i and $p(X)$ are relatively prime (from the definition of primitive polynomials), we have:

$$a_i(X) \neq 0 \quad (\text{A.24})$$

For $0 \leq i, j < 2^m - 1$ and $i \neq j$, we can show that: $a_i(X) \neq a_j(X)$. If they should be equal:

$$X^i + X^j = [q_i(X) + q_j(X)]p(X) + a_i(X) + a_j(X) = [q_i(X) + q_j(X)]p(X) \quad (\text{A.25})$$

It results that $p(X)$ should divide $X^i + X^j = X^i(1 + X^{j-i})$.

Since $p(X)$ and X^i are relatively prime, $p(X)$ should divide $1 + X^{j-i}$, which contradicts the definition of the prime polynomial $p(X)$, not to divide any polynomial with a smaller degree than $2^m - 1$, or $j - i \leq 2^m - 1$. The hypothesis is fake, so for any $0 \leq i, j < 2^m - 1$ and $i \neq j$ we must have:

$$a_i(X) \neq a_j(X) \quad (\text{A.26})$$

For $0, 1, \dots, 2^m - 2$, we obtain $2^m - 1$ non zero distinct polynomials $a_i(X)$ of degree $m-1$ or smaller.

Replacing X by α , in the relation (A.22), and taking into account the fact that $p(\alpha) = 0$, we obtain:

$$\alpha^i = a_i(\alpha) = a_{i0} + a_{i1} \alpha + \dots + a_{i(m-1)} \alpha^{m-1} \quad (\text{A.27})$$

The $2^m - 1$ non zero elements $\alpha^0, \alpha^1, \alpha^2, \dots, \alpha^{2^m-2}$ of F^* may be represented by $2^m - 1$ distinct non zero polynomials over $\mathbf{GF}(2)$ of degree $(m - 1)$ or smaller. The $\mathbf{0}$ element in F^* may be represented by the null polynomial.

In the following, we will define addition “+” on F^* :

$$\mathbf{0} + \mathbf{0} = \mathbf{0} \quad (\text{A.28})$$

and for $0 \leq i, j < 2^m - 1$

$$\mathbf{0} + \alpha^i = \alpha^i + \mathbf{0} = \alpha^i \quad (\text{A.29})$$

$$\alpha^i + \alpha^j = (a_{i0} + a_{j0}) + (a_{i1} + a_{j1})\alpha + \dots + (a_{i(m-1)} + a_{j(m-1)})\alpha^{m-1} \quad (\text{A.30})$$

the additions $a_{ie} + a_{je}$ being modulo-two summations.

From above, for $i = j$, it results $\alpha^i + \alpha^i = 0$ and for $i \neq j$, we have:

$$(a_{i0} + a_{j0}) + (a_{i1} + a_{j1})\alpha + \dots + (a_{i(m-1)} + a_{j(m-1)})\alpha^{m-1}, \text{ non zero.}$$

The relation (A.29) must be the polynomial expression for a certain α^k in F^* . So the set F is closed to addition operation “+”, previously defined.

We can immediately check that F^* is a commutative group for “+” operation. We notice that 0 is the additive identity; the addition modulo two being commutative and associative, the same thing happens for F^* . From (A.29) for $i = j$ we notice that the additive inverse (the opposite) is the element itself in F^* .

It was shown that $F^* = \{0 \ 1 \ \alpha \ \alpha^2 \ \dots \ \alpha^{2^m-2}\}$ is a commutative group for addition “+” and that the non zero elements in F^* form a commutative group for multiplication “•”. Using the polynomial representation for the elements in F^* and taking into account the fact that the polynomial multiplication satisfies the law of distributivity related to addition, it is easily shown that multiplication in F^* is distributive towards to addition in F^* .

So, the set F^* is a Galois field with 2^m elements, $\mathbf{GF}(2^m)$. All the addition and multiplication operations defined in $F^* = \mathbf{GF}(2^m)$ are done modulo two. It is thus noticed that (0, 1) form a subfield of $\mathbf{GF}(2^m)$, so $\mathbf{GF}(2)$ is a subfield of $\mathbf{GF}(2^m)$, the first one being called the basic field of $\mathbf{GF}(2^m)$. The characteristic of $\mathbf{GF}(2^m)$ is 2.

When constructing $\mathbf{GF}(2^m)$ from $\mathbf{GF}(2)$, we have developed two representations for the non zero elements in $\mathbf{GF}(2^m)$, an exponential representation and a polynomial one. The first one is convenient for multiplication, and the second one, for addition. There is also a third representation, matrix-type, as the following examples will show.

Remarks

In determining $\mathbf{GF}(2^m)$, we act as follows:

- we set the degree m of the primitive polynomial $p(X)$
- we calculate $2^m - 2$, which will give the maximum number of powers of α obtained from the primitive polynomial, after which this one is repeated $\alpha^{2^m-1} = 1$
- from the equation $p(\alpha) = 0$ we obtain α^m , after which any exponent is obtained from the previous one, taking into consideration the reduction $p(\alpha) = 0$

Example

We will determine the elements of $\mathbf{GF}(2^3)$, generated by the primitive polynomial $p(X) = 1 + X + X^3$.

We have $m = 3$ and $2^m - 2 = 6$, so $\alpha^7 = 1$, α being a root of $p(X)$ for which $1 + \alpha + \alpha^3 = 0$ so:

$$\begin{aligned} \alpha^3 &= 1 + \alpha \\ \alpha^4 &= \alpha + \alpha^2 \\ \alpha^5 &= \alpha^2 + \alpha^3 = 1 + \alpha + \alpha^2 \\ \alpha^6 &= \alpha + \alpha^2 + \alpha^3 = 1 + \alpha^2 \\ \alpha^7 &= \alpha + \alpha^3 = \alpha + 1 + \alpha = 1 \end{aligned}$$

For the matrix representation, we consider a linear matrix $(a_1 \ a_2 \ a_3)$ in which a_1, a_2, a_3 are the coefficients of the terms $\alpha^0, \alpha^1,$ and α^2 , respectively. So, for $\alpha^3 = 1 + \alpha$, we will have the matrix representation $(1 \ 1 \ 0)$. Similarly for the other exponents of α . The table below presents the elements of the field $GF(2^3)$, generated by the polynomial $p(X) = 1 + X + X^3$.

α power representation	Polynomial representation	Matrix representation
0	0	0 0 0
1	1	1 0 0
α	α	0 1 0
α^2	α^2	0 0 1
α^3	$1 + \alpha$	1 1 0
α^4	$\alpha + \alpha^2$	0 1 1
α^5	$1 + \alpha + \alpha^2$	1 1 1
α^6	$1 + \alpha^2$	1 0 1
α^7	1	1 0 0

Appendix A10 includes the tables for the representation of the fields $GF(2^3), GF(2^4), GF(2^5), GF(2^6)$.

A6 Basic Properties of Galois Fields, $GF(2^m)$

In the common algebra, we have seen that a polynomial with real coefficients does not have roots in the field of real numbers, but in that of complex numbers, which contains the previous one as a subfield. This observation is true as well for the polynomials with coefficients in $GF(2)$ which may not have roots from this one, but from an extension of the field $GF(2^m)$.

Example

The polynomial $X^4 + X^3 + 1$ is irreducible on $\mathbf{GF}(2)$, so it does not have roots in $\mathbf{GF}(2)$. Nevertheless, it has 4 roots in the extension $\mathbf{GF}(2^4)$, namely, by replacing the exponents of α (see A.10), in the polynomial, we find that $\alpha^4, \alpha^8, \alpha, \alpha^2$ are roots, so the polynomial can be written:

$$X^4 + X^3 + 1 = (X + \alpha)(X + \alpha^2)(X + \alpha^4)(X + \alpha^8)$$

Let now be $p(X)$, a polynomial with coefficients in $\mathbf{GF}(2)$. If β , an element in $\mathbf{GF}(2^m)$ is a root of $p(X)$, then we question whether $p(X)$ has other roots in $\mathbf{GF}(2^m)$ and what are those roots. The answer lies in the following property:

Property 1: Be $p(X)$, a polynomial with coefficients in $\mathbf{GF}(2)$. Be β an element of the extension of $\mathbf{GF}(2)$. If β is a root of $p(X)$, then for any $l \geq 0, \beta^{2^l}$ is also a root of $p(X)$.

This is easily demonstrated taking into account relation (A.14):

$$[p(X)]^{2^l} = p(X^{2^l}) \text{ by replacing } X \text{ by } \beta \text{ we have:}$$

$$[p(\beta)]^{2^l} = p(\beta^{2^l})$$

So, if $p(\beta) = 0$, it results that and so $p(\beta^{2^l}) = 0$ so β^{2^l} is also root of $p(X)$. This can be easily noticed from the previous example. The element β^{2^i} is called the conjugate of β . Property 1 says that if β is an element in $\mathbf{GF}(2^m)$ and a root of the polynomial $p(X)$ in $\mathbf{GF}(2)$, then all the distinct conjugates of β , elements of $\mathbf{GF}(2^m)$, are roots of $p(X)$.

For example, the polynomial $p(X) = 1 + X^3 + X^4 + X^5 + X^6$ has α^4 , as root in $\mathbf{GF}(2^4)$:

$$\begin{aligned} p(\alpha^4) &= 1 + (\alpha^3)^4 + (\alpha^4)^4 + (\alpha^5)^4 + (\alpha^6)^4 = 1 + \alpha^{12} + \alpha^{16} + \alpha^{20} + \alpha^{24} = \\ &= 1 + \alpha^{12} + \alpha + \alpha^5 + \alpha^9 = 1 + (1 + \alpha + \alpha^2 + \alpha^9) + \alpha + (\alpha + \alpha^2) + (\alpha + \alpha^3) = 0 \end{aligned}$$

The conjugates of α^4 are:

$$(\alpha^4)^2 = \alpha^8, (\alpha^4)^{2^2} = \alpha^{16} = \alpha, (\alpha^4)^{2^3} = \alpha^{32} = \alpha^2$$

We note that $(\alpha^4)^2 = \alpha^{64} = \alpha^0$, so if we go on, we shall see that the values found above repeat. It results, according to property 1, that α^8 , α and α^2 must also be roots of $p(X) = 1 + X^3 + X^4 + X^5 + X^6$.

Similarly, the same polynomial has the roots α^5 as well, because indeed $p(\alpha^5) = 1 + \alpha^{15} + \alpha^{20} + \alpha^{25} + \alpha^{30} = 1 + 1 + \alpha^5 + \alpha^{10} + 1 = 1 + \alpha + \alpha^2 + 1 + \alpha + \alpha^2 = 0$

So, this one and its conjugates $(\alpha^5)^2 = \alpha^{10}$, $(\alpha^5)^2 = \alpha^{20} = \alpha^5$ are roots of $p(X) = 1 + X^3 + X^4 + X^5 + X^6$.

In this way we have obtained all the 6 roots of $p(X)$: α^4 , α^8 , α , α^2 , α^5 , α^{10} .

If β is a non zero element in the field $\mathbf{GF}(2^m)$, we have $\beta^{2^m-1} = 1$, therefore $\beta^{2^m-1} + 1 = 0$, so β is a root of $X^{2^m-1} + 1$. It follows that any non zero element from $\mathbf{GF}(2^m)$ is root of $X^{2^m-1} + 1$. As its degree is $2^m - 1$, it results that the $2^m - 1$ non zero elements of $\mathbf{GF}(2^m)$ are all roots of $X^{2^m-1} + 1$.

The results lead to:

Property 2: The $2^m - 1$ non zero elements of $\mathbf{GF}(2^m)$ are all roots of $X^{2^m-1} + 1$, or, all the elements of $\mathbf{GF}(2^m)$ are all roots of the polynomial $X^{2^m} + X$.

Since any β element in $\mathbf{GF}(2^m)$ is root of the polynomial $X^{2^m} + X$, β can be root of a polynomial on $\mathbf{GF}(2)$, the degree of which should be smaller than 2^m . Be $\Phi(X)$ the smallest degree polynomial on $\mathbf{GF}(2)$, so that $\Phi(\beta) = 0$. This polynomial is called *minimal polynomial* of β .

Example

The minimal polynomial of the zero element in $\mathbf{GF}(2^m)$ is X , and that of the unit element 1, is $X + 1$.

Further on, we will demonstrate a number of *properties of the minimal polynomials*.

- *The minimal polynomial $\Phi(X)$ of an element of the field is irreducible.*

We suppose that $\Phi(X)$ is not irreducible and that $\Phi(X) = \Phi_1(X) \cdot \Phi_2(X) \Rightarrow \Phi(\beta) = \Phi_1(\beta) \cdot \Phi_2(\beta)$, so either $\Phi_1(\beta) = 0$, or

$\Phi_2(\beta) = 0$, which contradicts the hypothesis that $\Phi(X)$ is the polynomial with the smallest degree for which $\Phi(\beta) = 0$. It results that the minimal polynomial $\Phi(X)$ is irreducible.

- Be $p(X)$ a polynomial on $\mathbf{GF}(2)$, and $\Phi(X)$ the minimal polynomial of the element β in the field.

As β is a root of $p(X)$, then $p(X)$ is divisible by $\Phi(X)$.

After division, it results: $p(X) = a(X)\Phi(X) + r(X)$, where the remainder degree is smaller than the degree of $\Phi(X)$. After replacing β in the above relation, we will obtain:

$$p(\beta) = a(\beta) \cdot \Phi(\beta) + r(\beta), \quad p(\beta) = 0, \quad \Phi(\beta) = 0 \Rightarrow r(\beta) = 0$$

The remainder being zero, it results that $\Phi(X)$ divides $p(X)$.

- The minimal polynomial $\Phi(X)$ of an element β in $\mathbf{GF}(2^m)$ divides $X^{2^m} + X$ meaning all the roots of $\Phi(X)$ are in $\mathbf{GF}(2^m)$.

Property 3: Let β an element in $\mathbf{GF}(2^m)$, and e the smallest non zero integer so that $\beta^{2^e} = \beta$. Then:

$$p(x) = \prod_{i=0}^{e-1} (x + \beta^{2^i}) \tag{A.31}$$

is an irreducible polynomial on $\mathbf{GF}(2)$.

In order to demonstrate this property, we consider:

$$\begin{aligned} [p(x)]^2 &= \left[\prod_{i=0}^{e-1} (x + \beta^{2^i}) \right]^2 = \prod_{i=0}^{e-1} (x + \beta^{2^i})^2 = \prod_{i=0}^{e-1} (x^2 + \beta^{2^{i+1}}) = \prod_{i=1}^e (x^2 + \beta^{2^i}) = \\ &= \left[\prod_{i=1}^{e-1} (x^2 + \beta^{2^i}) \right] (x^2 + \beta^{2^e}) \end{aligned}$$

As $\beta^{2^e} = \beta$, we have:

$$[p(x)]^2 = \prod_{i=0}^{e-1} (x^2 + \beta^{2^i}) = p(x^2) = \sum_{i=0}^e p_i x^{2i} \tag{A.32}$$

Let the polynomial $p(X) = p_0 + p_1X + \dots + p_eX^e$, $p_e = 1$. Taking A.32 into account, we have:

$$[p(X)]^2 = [p_0 + p_1X + \dots + p_eX^e]^2 = \sum_{i=0}^e p_i^2 X^{2i} \tag{A.33}$$

From the relations (A.32) and (A.33), we obtain:

$$\sum_{i=0}^e p_i X^{2i} = \sum_{i=0}^e p_i^2 X^{2i} \quad (\text{A.34})$$

In order to have the equality, we must have:

$$p_i = p_i^2, \text{ so } p_i = 0 \text{ or } 1.$$

So it is compulsory that $p(X)$ has the coefficients in $\mathbf{GF}(2)$.

We suppose that $p(X)$ would not be irreducible on $\mathbf{GF}(2)$ and that $p(X) = a(X) \cdot b(X)$. If $p(\beta) = 0$, then automatically $a(\beta) = 0$ or $b(\beta) = 0$.

If $a(\beta) = 0$, $a(X)$ has the roots $\beta, \beta^2, \dots, \beta^{2^e-1}$ so its degree is e and $p(X) = a(X)$. Similarly for $b(X)$, so $p(X)$ is irreducible. A direct consequence of the last two properties is the following property:

Property 4: Let $\Phi(X)$ the minimal polynomial of the element β in $\mathbf{GF}(2^m)$. Let e be the smallest integer so that $\beta^{2^e} = \beta$. Then we have:

$$\Phi(X) = \prod_{i=0}^{e-1} (X + \beta^{2^i}) \quad (\text{A.35})$$

Examples

1. Let be the Galois field $\mathbf{GF}(2^4)$, given in Appendix A10.

Let $\beta = \alpha^3$. The conjugates of β are:

$$\beta^2 = \alpha^6, \beta^{2^2} = \alpha^{12}, \beta^{2^3} = \alpha^{24} = \alpha^9$$

The minimal polynomial of $\beta = \alpha^3$ is:

$$\begin{aligned} \Phi(X) &= (X + \alpha^3)(X + \alpha^6)(X + \alpha^{12})(X + \alpha^9) = (X^2 + \alpha^2 X + \alpha^9)(X^2 + \alpha^8 X + \alpha^6) = \\ &= X^4 + X^3 + X^2 + X + 1 \end{aligned}$$

There is also another possibility of obtaining the minimal polynomial of an element in the field, as we shall see further on:

2. We want to find the minimal polynomial, $\Phi(X)$, of the element $\beta = \alpha^7$ in $\mathbf{GF}(2^4)$ from A10.

The distinct conjugates are:

$$\beta^2 = \alpha^{14}, \beta^{2^2} = \alpha^{28} = \alpha^{13}, \beta^{2^3} = \alpha^{56} = \alpha^{11}$$

As $\Phi(X)$ must be of 4th degree, it must have the form:

$$\Phi(X) = a_0 + a_1X + a_2X^2 + a_3X^3 + X^4$$

Replacing X by β we obtain:

$$\Phi(\beta) = a_0 + a_1\beta + a_2\beta^2 + a_3\beta^3 + \beta^4 = 0$$

Using the polynomial representations for β, β^2, β^3 and β^4 we obtain:

$$\begin{aligned} a_0 + a_1(\mathbf{1} + \alpha + \alpha^3) + a_2(\mathbf{1} + \alpha^3) + a_3(\alpha^2 + \alpha^3) + (\mathbf{1} + \alpha + \alpha^3) &= 0 \\ (a_0 + a_1 + a_2 + \mathbf{1}) + a_1\alpha + (a_3 + \mathbf{1})\alpha^2 + (a_1 + a_2 + a_3 + \mathbf{1})\alpha^3 &= 0 \end{aligned}$$

All coefficients must be zero:

$$\begin{cases} a_0 + a_1 + a_2 + 1 = 0 \\ a_1 = 0 \\ a_3 + 1 = 0 \\ a_1 + a_2 + a_3 + 1 = 0 \end{cases} \Rightarrow \begin{cases} a_0 = 1 \\ a_1 = a_2 = 0 \\ a_3 = 1 \end{cases}$$

So for $\beta = \alpha^7$ the minimal polynomial is:

$$\Phi(X) = 1 + X^3 + X^4$$

In what follows we shall present tables of minimal polynomials in $\mathbf{GF}(2^m)$ for $m = 3, m = 4$ and $m = 5$.

Table A.1 Minimal polynomials for GF(2³) and generating polynomial 1 + X+ X³

Conjugated Roots	Minimal Polynomials
0	0
1	X
$\alpha, \alpha^2, \alpha^4$	$X^3 + X + 1$
$\alpha^3, \alpha^6, \alpha^{12}, \alpha^5$	$X^3 + X^2 + 1$

Table A.2 Minimal polynomials for GF(2⁴) and generating polynomial 1 + X+ X⁴

Conjugated Roots	Minimal Polynomials
0	0
1	X
$\alpha, \alpha^2, \alpha^4, \alpha^8$	$X^4 + X + 1$
$\alpha^3, \alpha^6, \alpha^9, \alpha^{12}$	$X^4 + X^3 + X^2 + 1$
α^5, α^{10}	$X^2 + X + 1$
$\alpha^7, \alpha^{11}, \alpha^{13}, \alpha^{14}$	$X^4 + X^3 + 1$

Table A.3 Minimal polynomials for $\text{GF}(2^5)$ and generating polynomial $1 + X + X^5$

Conjugated Roots	Minimal Polynomials
0	0
1	X
$\alpha, \alpha^2, \alpha^4, \alpha^8, \alpha^{16}$	$X^5 + X^2 + 1$
$\alpha^3, \alpha^6, \alpha^{12}, \alpha^{24}, \alpha^{48} = \alpha^{17}$	$X^5 + X^4 + X^3 + X^2 + 1$
$\alpha^5, \alpha^{10}, \alpha^{20}, \alpha^{40} = \alpha^9, \alpha^{18}$	$X^5 + X^4 + X^2 + X + 1$
$\alpha^7, \alpha^{14}, \alpha^{28}, \alpha^{56} = \alpha^{25}, \alpha^{50} = \alpha^{21}$	$X^5 + X^3 + X^2 + X + 1$
$\alpha^{15}, \alpha^{30}, \alpha^{60} = \alpha^{29}, \alpha^{58} = \alpha^{28}, \alpha^{54} = \alpha^{23}$	$X^5 + X^3 + 1$
$\alpha^{11}, \alpha^{22}, \alpha^{44} = \alpha^{13}, \alpha^{26}, \alpha^{52} = \alpha^{21}$	$X^5 + X^4 + X^3 + X + 1$

Some explanations for Table A.2: α being root of the polynomial $p(X) = X^4 + X + 1$, it has the conjugates $\alpha^2, \alpha^4, \alpha^8$ which are also roots, so for all of them, the minimal polynomial is : $X^4 + X + 1$.

Among the exponents of α , given above, the smallest one that did not appear is α^3 , which has the conjugates $\alpha^6, \alpha^{12}, \alpha^{24} = \alpha^9$ and for all of them, the minimal polynomial is $X^4 + X^3 + X^2 + X + 1$. For α^5 we have only the conjugate α^{10} as α^{20} . The corresponding minimal polynomial is $X^2 + X + 1$. Finally, for α^7 we have the conjugates $\alpha^{11}, \alpha^{13}, \alpha^{14}$ to which corresponds the minimal polynomial $X^4 + X^3 + 1$.

In order to find the minimal polynomial for root α , we have assumed that the other roots are exponents of α . The justification consists in that the primitive polynomial divides $X^{2^m} + 1$.

But if m is not prime, each roots of the binomial equation (except 1), raised to $m-1$ powers repeat all the others. When it is not prime ($m = 4$), some of the minimal polynomials can have smaller degrees than the primitive polynomial. As the tables for $m = 3$ and $m = 5$ (prime numbers) show, the minimal polynomials are the primitive polynomials of degree 3 and 5.

Since the two polynomials in $\text{GF}(2^m)$ cannot have a common root (because they would coincide), the minimal polynomials must be prime pairs. It results that the $2^m - 1$ roots must be distributed among m degree polynomials or even smaller degree ones. Thus, if $m = 4$ $2^4 - 1 = 15$, it results that there will be three 4th degree polynomials, one 2nd degree polynomial and one first degree polynomial. For $m = 3$ $2^3 - 1 = 7$, we will have two third degree polynomials and one first degree polynomial. For $m = 5$ $2^5 - 1 = 31$, there will be six fifth degree polynomials and one first degree polynomial.

Property 5: It is a direct consequence of the previous one and stipulates that if $\Phi(X)$ is the minimal polynomial of an element β in $\mathbf{GF}(2^m)$ and if e is the degree of $\Phi(X)$, then e is the smallest integer such that $\beta^{2^e} = \beta$. Obviously, $e \leq m$.

In particular, it can be shown that the minimal polynomial degree of each element in $\mathbf{GF}(2^m)$ divides m . The tables prove this affirmation.

When constructing the Galois field $\mathbf{GF}(2^m)$, we use a minimal polynomial $p(X)$ of m degree and we have the element α which is $p(X)$ root. As the exponents of α generate all the non zero elements of $\mathbf{GF}(2^m)$, α is a primitive element.

All the conjugates of α are primitive elements of $\mathbf{GF}(2^m)$. In order to see this, let n be the order of α^{2^1} , $1 > 0$, and we have:

$$\left(\alpha^{2^1}\right)^n = \alpha^{n \cdot 2^1} = \mathbf{1} \tag{A.36}$$

As α is a primitive element of $\mathbf{GF}(2^m)$, its order is $2^m - 1$. From the above relation, it results that 2^1 must be divisible to $2^m - 1$, $n = q(2^m - 1)$. But $n = k(2^m - 1)$; $n = 2^m - 1$, so α^2 is a primitive element of $\mathbf{GF}(2^m)$.

Generally, if β is a primitive element of $\mathbf{GF}(2^m)$, all its conjugates $\beta^2, \beta^{2^e}, \dots$ are also primitive elements of $\mathbf{GF}(2^m)$.

Using the tables for $\mathbf{GF}(2^m)$, linear equation systems can be solved. Be the following system:

$$\begin{cases} X + \alpha^7 Y = \alpha^2 \\ \alpha^{12} X + \alpha^8 Y = \alpha^4 \mid \alpha^3 \text{ (which is the inverse of } \alpha^{12}) \end{cases}$$

$$\begin{cases} X + \alpha^7 Y = \alpha^2 \\ X + \alpha^{11} Y = \alpha^7 \end{cases}$$

By addition and expressing α^{11}, α^7 and reducing the terms, we obtain:

$$(1 + \alpha^2) Y = \mathbf{1} + \alpha + \alpha^2 + \alpha^3, \text{ but } \alpha^8 = \mathbf{1} + \alpha^2 \text{ and the inverse of } \alpha^8 \text{ is } \alpha^7$$

So, multiplying this equation by α^7 , we obtain:

$$Y = \alpha^7 + \alpha^8 + \alpha^{3+7} + \alpha^9 = \mathbf{1} + \alpha = \alpha^4$$

It follows that $Y = \alpha^4$.

Similarly for $X = \alpha^9$.

If we want to solve the equation $f(X) = X^2 + \alpha^7 X + \alpha = \mathbf{0}$, we cannot do this by the regular formula, as we are working in modulo 2. Then, if $f(X) = 0$, it has a solution in $\mathbf{GF}(2^4)$; the solution is obtained by replacing X by all the elements from A10. We find $f(\alpha^6) = 0$ and $f(\alpha^{10}) = 0$, so α^6 and α^{10} are roots.

A7 Matrices and Linear Equation Systems

A *matrix* is a table of elements for which two operations were defined: addition and multiplication. If we have m lines and n columns, the matrix is called a $m \times n$ matrix.

Two *matrices are equal* when all the corresponding elements are equal. *Matrix addition* is done only for matrixes having the same dimensions, and the result is obtained by adding the corresponding elements. The null matrix is the matrix with all elements zero.

The set of all matrices with identical dimensions forms a commutative group related to addition.

The null element is the null matrix.

Particular cases of matrices are the linear matrix $[a \ b \ \dots]$ and column

$$\text{matrix} \begin{bmatrix} a \\ b \\ \vdots \end{bmatrix}.$$

Multiplication of two matrices cannot be done unless the columns number of the first matrix equals the lines number of the second one. Multiplication is not commutative. For square matrixes, $m \times m$ of degree m , we have the unit matrix that has all elements null, except those ones on the main diagonal which equal 1. Its determinant can be defined, according to the known rule, only for these ones. For such a matrix, we have the inverse matrix only when its determinant is not null.

For any *matrix* the notion of *rank* is defined as follows: by suppressing lines and columns in the given matrix and keeping the order of the elements left, we obtain determinants of orders from $\min(n, m)$ to 1. From these ones, the maximum order of non zero determinants is the matrix rank.

There are three operations that change the matrix elements, but maintain its rank, meaning: switching lines with columns, multiplying a line (column) by a non zero number, adding the elements of a line (column) to another line (column).

These operations allow us to get a matrix for which, on each line and column we have one non zero element the most. The number of these elements gives us the matrix rank.

Example: Calculate the rank of the following matrix:

$$\begin{pmatrix} 1 & 2 & 2 & 4 \\ 4 & 5 & 8 & 10 \\ 3 & 1 & 6 & 2 \end{pmatrix} \sim \begin{pmatrix} 1 & 2 & 2 & 4 \\ 0 & -3 & 0 & -6 \\ 0 & -5 & 0 & -10 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -3 & 0 & -6 \\ 0 & -5 & 0 & -10 \end{pmatrix} \sim \\ \sim \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 0 & 2 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

We have two non zero elements, so the rank of the matrix is 2.

Linear equations systems

Be the system:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \dots\dots\dots\dots\dots\dots\dots\dots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n = b_2 \end{cases} \quad (A.37)$$

For such a system, the rank r of the matrix $m \times n$ corresponding to the coefficients of the unknowns x_1, \dots, x_n . At the same time, we choose a non zero determinant of order r , which is called *principal determinant*. The equations and the unknowns that give the elements in this determinant are called *principal equations*, and the others, secondary. For each secondary equation, a characteristic determinant is set up, by bounding the principal determinant by a line and a column. The line contains the coefficients of the main unknowns, and the column, the free terms of the principal and secondary equations.

We have *Rouche theorem*, which says that the *system* is *compatible* (it has solutions) if and only if all the characteristic determinants are null, the solutions being obtained by Cramer rule. Secondary unknowns are considered parameters, case in which we have an infinite number of solutions.

If the rank r equals the number n of the unknowns we have a *unique solution*; on the contrary, we have secondary unknowns, so an infinite number of solutions.

If $n=m$ and the rank is n , the system is called *Cramer system* and there is a rule for expressing the solution using determinants. But, as calculating the determinants involve a high volume of operations, in application we use *Gauss algorithm*. It starts from the extended matrix of the equation system (the coefficients matrix of the unknowns to which the column of free terms is added), on which the described operations for determining the rank of a matrix are clearly made, working

only on lines. Thus the operations number to be solved is much smaller than the one required for calculating the determinants. A simple example will illustrate this method which can be applied to any binary field.

Be the system:

$$\begin{cases} x + 2y + 3z = -3 \\ 2x - y + z = -1 \\ -x + y + 2z = -4 \end{cases}$$

We have:

$$\begin{aligned} & \begin{pmatrix} (1) & 2 & 3 & -3 \\ 2 & -1 & 1 & -1 \\ -1 & 1 & 2 & -4 \end{pmatrix} \sim \begin{pmatrix} (1) & 2 & 3 & -3 \\ 0 & -5 & -5 & -5 \\ 0 & 3 & 5 & -7 \end{pmatrix} \sim \begin{pmatrix} (1) & 2 & 3 & -3 \\ 0 & (1) & 1 & -1 \\ 0 & 3 & 5 & -7 \end{pmatrix} \sim \\ & \sim \begin{pmatrix} (1) & 0 & 1 & -1 \\ 0 & (1) & 1 & -1 \\ 0 & 0 & (2) & -4 \end{pmatrix} \sim \begin{pmatrix} (1) & 0 & 1 & -1 \\ 0 & (1) & 1 & -1 \\ 0 & 0 & (1) & -2 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -2 \end{pmatrix} \end{aligned}$$

The solution is $x = 1, y = 1, z = 2$.

A8 Vector Spaces

A8.1 Defining Vector Space

Let $(V, +)$ be an Abelian group, with elements called *vectors* and denoted \bar{v}, \bar{x}, \dots . Let $(F, +, \cdot)$ be a field, with elements called *scalars*, having as neutral elements 0 and 1.

We call *vector space* over field F , the Abelian group $(V, +)$, on which an outer composition law with operators in F is given, called the multiplication of vectors by scalars: $F \times V \rightarrow V, \forall a \in F \text{ and } \bar{u} \in V$ and which satisfies the axioms:

$$\begin{aligned} (a + b)\bar{u} &= a\bar{u} + b\bar{u} \\ a(\bar{u} + \bar{v}) &= a\bar{u} + a\bar{v} \\ (ab)\bar{u} &= a(b\bar{u}) \\ 1\bar{v} &= \bar{v} \\ 0\bar{v} &= \bar{0} \end{aligned} \tag{A.38}$$

Example

The space vectors and the real numbers form a vector space. The set of vectors in the three dimension space is determined by three coordinates, which can be assembled in a linear matrix:

$$\bar{v} = (x, y, z).$$

The set of linear matrices, with a dimension of 1×3 , forms a vector space. More general, the set of matrices with a dimension of $1 \times n$ forms a vector space.

A8.2 Linear Dependency and Independence

Be the vectors $\bar{v}_1, \bar{v}_2, \dots, \bar{v}_n$ in the vector space $F \times V$ and the scalars $\lambda_1, \lambda_2, \dots, \lambda_n$. Any expression having the form $\lambda_1 \bar{v}_1, \lambda_2 \bar{v}_2, \dots, \lambda_n \bar{v}_n$ is called *linear combination* of the respective *vectors*. If this combination is zero, not all of the scalars being null, then:

$$\lambda_1 \bar{v}_1 + \lambda_2 \bar{v}_2, \dots + \lambda_n \bar{v}_n = 0 \quad (\text{A.39})$$

we say that the *vectors*: $\bar{v}_1, \bar{v}_2, \dots, \bar{v}_n$ are *linear independent*. In the opposite situation, they are called *linear dependent*.

Thus, two vectors in the plane can be linear independent, but three cannot. The maximum number of linear independent vectors in a vector space is called *space dimension*, and any system of linear independent vectors forms a *space base*; all the other space vectors can be expressed with the base ones. So, if the base vectors are noted by $\bar{e}_1, \bar{e}_2, \dots, \bar{e}_n$, any vector from that space can be written as:

$$\bar{V} = \alpha_1 \bar{e}_1 + \alpha_2 \bar{e}_2 + \dots + \alpha_n \bar{e}_n \quad (\text{A.40})$$

in which the numbers $\alpha_1, \alpha_2, \dots, \alpha_n$ are called *vector coordinates*.

If the space dimension is n , then the component matrix of the n vectors that form a base has the rank n .

In an n -dimension space we can always have a number $m < n$ of vectors linear independent. Their linear combinations generate only some of the space vectors, so they form a *subspace* of the given space, the dimension of which equals the matrix rank of their components.

Coming back to the Galois field, the vector space on $\mathbf{GF}(2)$ plays a central part in the coding theory. We consider the string (a_1, a_2, \dots, a_n) , in which each component a_i is an element of the binary field $\mathbf{GF}(2)$. This string is called *n -tuple on $\mathbf{GF}(2)$* . As each element can be 0 or 1, it results that we can set up distinct n -tuples, which form the set V_n . We define addition "+" on V_n , for

$\bar{u} = (u_0, u_1, \dots, u_{n-1})$, $\bar{v} = (v_0, v_1, \dots, v_{n-1})$, $\bar{u}, \bar{v} \in V_n$ by the relation:

$$\bar{u} + \bar{v} = (u_0 + v_0, u_1 + v_1, \dots, u_{n-1} + v_{n-1}) \quad (\text{A.41})$$

the additions being made modulo two.

It is obvious that $\bar{u} + \bar{v}$ is an n -tuple on $\mathbf{GF}(2)$ as well. As V_n is closed to addition, we will easily check that V_n is a commutative group to the above defined addition.

Taking into account that addition modulo two is commutative and associative, it results the same thing for the above defined addition. Having in view that $\bar{v} = (0, 0, \dots, 0)$ is the additive identity and that:

$$\bar{v} + \bar{v} = (v_0 + v_0, v_1 + v_1, \dots, v_{n-1} + v_{n-1}) = (0, 0, \dots, 0) = \bar{0},$$

the inverse of each elements being itself, V_n is a commutative group to the above defined addition.

We define the *scalar multiplication* of an n -tuple, \bar{v} in V_n with an element a in $\mathbf{GF}(2)$, by: $a(v_0, v_1, \dots, v_{n-1}) = (a v_0, a v_1, \dots, a v_{n-1})$ modulo two multiplication.

If $a = 1$ we have: $a\bar{v} = \bar{v}$. It is easy to see that the two identical operations defined above satisfy the distributivity and associativity laws. And the set V_n of n -tuples on $\mathbf{GF}(2)$ forms a vector space over $\mathbf{GF}(2)$.

A8.3 Vector Space

V being a vector space on field F , it may happen that a subset S of V be vector space on F as well. This one will be called a *subspace of V* .

So, if S is a non zero subset of space V in the field F , S is subspace of V if:

1. For any two vectors \bar{u} and \bar{v} in S , $\bar{u} + \bar{v}$ must be in S .
2. For any element $a \in F$ and any vector in S , $a\bar{u}$ must be in S .

Or, if we have the vector set $\bar{v}_1, \bar{v}_2, \dots, \bar{v}_k$ in the space V , then the set of all linear combinations of these vectors form a subspace of V .

We consider the vector space V_n of all n -tuples on $\mathbf{GF}(2)$. We form the following n -tuples:

$$\begin{aligned} \bar{e}_0 &= (1, 0, \dots, 0) \\ \bar{e}_1 &= (0, 1, \dots, 0) \\ &\dots \\ \bar{e}_{n-1} &= (0, 0, \dots, 1) \end{aligned} \quad (\text{A.42})$$

Then any n -tuples $(a_0, a_1, \dots, a_{n-1})$ in V_n can be expressed with these ones; it follows that vectors (A.42) generate the entire space V_n of n -tuples over $\mathbf{GF}(2)$.

They are linear independent and thus form a base of V_n and its dimension is n . If $k < n$, the linear independent vectors $\overline{v_1}, \overline{v_2}, \dots, \overline{v_k}$ generate a subspace of V_n .

Be $\overline{u} = (u_0, u_1, \dots, u_{n-1})$, $\overline{v} = (v_0, v_1, \dots, v_{n-1})$, $\overline{u}, \overline{v} \in V_n$. By *inner product / dot product* of \overline{u} and \overline{v} we understand the scalar:

$$\overline{u} \overline{v} = u_0 v_0 + u_1 v_1 + \dots + u_{n-1} v_{n-1} \quad (\text{A.42})$$

all calculated in modulo 2. If $\overline{u} \overline{v} = 0$, \overline{u} and \overline{v} are orthogonal.

The *inner product / dot product* has the properties:

$$\overline{u} \overline{v} = \overline{v} \overline{u} \quad (\text{A.44})$$

$$\overline{u}(\overline{v} + \overline{w}) = \overline{u} \overline{v} + \overline{u} \overline{w} \quad (\text{A.45})$$

$$(\overline{a} \overline{u}) \overline{v} = \overline{a}(\overline{u} \overline{v}) \quad (\text{A.46})$$

Let be S a subspace of dimension k over V_n and S_d the vector set in V_n , such that for $\forall \overline{u} \in S$ and $\overline{v} \in S_d$ we have

$$\overline{u} \overline{v} = 0 \quad (\text{A.47})$$

For any element a in $\mathbf{GF}(2)$ and any $\overline{v} \in S_d$ we have:

$$\overline{a} \overline{v} = \begin{cases} \overline{0} & \text{if } a = 0 \\ \overline{v} & \text{if } a = 1 \end{cases} \quad (\text{A.48})$$

It follows that $\overline{a} \overline{v}$ is also in S_d .

Let \overline{v} and \overline{w} two vectors in S_d . For any vector $\overline{u} \in S$ we have:

$$\overline{u}(\overline{v} + \overline{w}) = \overline{u} \overline{v} + \overline{u} \overline{w} = 0 + 0 = 0 \quad (\text{A.49})$$

This means that if \overline{v} and \overline{w} are orthogonal with \overline{u} , the sum vector $\overline{v} + \overline{w}$ is also orthogonal with \overline{u} , so $\overline{v} + \overline{w}$ is a vector in S_d . So S_d is also a vector space, besides it is a subspace of V_n . This subspace, S_d , is called *dual space* (or *null*) of S . Its dimension is $n - k$, where n is the dimension of the space V_n , and k the dimension of the space S :

$$\dim(S) + \dim(S_d) = n \quad (\text{A.50})$$

In order to determine the space, and the dual subspace, we look for the base of orthogonal vectors. Only those vectors which are orthogonal with all the subspace vectors from which we have started, are selected as base of the dual space.

Example

From the set of 32 elements of 5-tuples (a_1, \dots, a_5) , we consider 7 of them and we look for the space dimension that they form. We have:

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \sim \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \sim \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \sim$$

$$\sim \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \sim \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

It follows that the rank is 3 and a base is formed by the vectors that correspond to the lines that contain 1: (11100), (01010), (10001).

Now we look for the orthogonal vectors in the subspace considered and we use them to form the subspace.

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \sim \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \sim \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \sim$$

$$\sim \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \sim \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

We get that the space dimension is 4, and one of its bases is: (10101), (01110), (10010), (00100) and (00000).

A9 Table for Primitive Polynomials of Degree k (k max = 100)

Remark

The table lists only one primitive polynomial for each degree $k \leq 100$. In this table only the degrees of the terms in the polynomial are given; thus 7 1 0 stands for $x^7 + x + 1$.

1	0				51	6	3	1	0		
2	1	0			52	3	0				
3	1	0			53	6	2	1	0		
4	1	0			54	6	5	4	3	2	0
5	2	0			55	6	2	1	0		
6	1	0			56	7	4	2	0		
7	1	0			57	5	3	2	0		
8	4	3	2	0	58	6	5	1	0		
9	4	0			59	6	5	4	3	1	0
10	3	0			60	1	0				
11	2	0			61	5	2	1	0		
12	6	4	1	0	62	5	3	0			
13	4	3	1	0	63	1	0				
14	5	3	1	0	64	4	3	1	0		
15	1	0			65	4	3	1	0		
16	5	3	2	0	66	8	6	5	3	2	0
17	3	0			67	5	2	1	0		
18	5	2	1	0	68	7	5	1	0		
19	5	2	1	0	69	6	5	2	0		
20	3	0			70	5	3	1	0		
21	2	0			71	5	3	1	0		
22	1	0			72	6	4	3	2	1	0
23	5	0			73	4	3	2	0		
24	4	3	1	0	74	7	4	3	0		
25	3	0			75	6	3	1	0		
26	6	2	1	0	76	5	4	2	0		
27	5	2	1	0	77	6	5	2	0		
28	3	0			78	7	2	1	0		
29	2	0			79	4	3	2	0		
30	6	4	1	0	80	7	5	3	2	1	0
31	3	0			81	4	0				

32	7	5	3	2	1	0	82	8	7	6	4	1	0
33	6	4	1	0			83	7	4	2	0		
34	7	6	5	2	1	0	84	8	7	5	3	1	0
35	2	0					85	8	2	1	0		
36	6	5	4	2	1	0	86	6	5	2	0		
37	5	4	3	2	1	0	87	7	5	1	0		
38	6	5	1	0			88	8	5	4	3	1	0
39	4	0					89	6	5	3	0		
40	5	4	3	0			90	5	3	2	0		
41	3	0					91	7	6	5	3	2	0
42	5	4	3	2	1	0	92	6	5	2	0		
43	6	4	3	0			93	2	0				
44	6	5	2	0			94	6	5	1	0		
45	4	3	1	0			95	6	5	4	2	1	0
46	8	5	3	2	1	0	96	7	6	4	3	2	0
47	5	0					97	6	0				
48	7	5	4	2	1	0	98	7	4	3	2	1	0
49	6	5	4	0			99	7	5	4	0		
50	4	3	2	0			100	8	7	2	0		

A10 Representative Tables for Galois Fields $GF(2^k)$

Remark

The tables list the powers of α and the matrix representation for Galois Fields $GF(2^k)$, $k \leq 6$. For example, in the first table 3 1 1 0 stands for $\alpha^3 = 1 + \alpha$.

1. $GF(2^3)$ generated by $p(x) = 1+x+x^3$

-	000
0	100
1	010
2	001
3	110
4	011
5	111
6	101

2. $GF(2^4)$ generated by $p(x) = 1+x+x^4$

-	0000	8	1010
0	1000	9	0101
1	0100	10	1110
2	0010	11	0111
3	0001	12	1111

4	1100	13	1011
5	0110	14	1001
6	0011		
7	1101		

3. $\text{GF}(2^5)$ generated by $\mathbf{p(x) = 1+x^2+x^5}$

-	00000	10	10001	21	00011
0	10000	11	11100	22	10101
1	01000	12	01110	23	11110
2	00100	13	00111	24	01111
3	00010	14	10111	25	10011
4	00001	15	11111	26	11101
5	10100	16	11011	27	11010
6	01010	17	11001	28	01101
7	00101	18	11000	29	10010
8	10110	19	01100	30	01001
9	01011	20	00110		

4. $\text{GF}(2^6)$ generated by $\mathbf{p(x) = 1+x+x^6}$

-	000000	21	101011	43	111011
0	100000	22	100101	44	101101
1	010000	23	100101	45	100110
2	001000	24	100010	46	010011
3	000100	25	010001	47	111001
4	000010	26	111000	48	101100
5	000001	27	011100	49	010110
6	110000	28	001110	50	001011
7	011000	29	000111	51	110101
8	001100	30	110011	52	101010
9	000110	31	101001	53	010101
10	000011	32	100100	54	111010
11	110001	33	010010	55	011101
12	101000	34	001001	56	111110
13	010100	35	110100	57	011111
14	001010	36	011010	58	111111
15	000101	37	001101	59	101111
16	110010	38	110110	60	100111
17	011001	39	011011	61	100011
18	111100	40	111101	62	100001
19	011110	41	101110		
20	001111	42	010111		

A11 Tables of the Generator Polynomials for BCH Codes

Remark

The table lists the corresponding generator polynomial coefficients for BCH codes of different length $n \leq 127$, with different correctable errors (t). The coefficients are written in octal.

For example for a BCH(15,7) code, from the table results: $n = 15$, $m = 7$, $t = 2$ and $g(x) =: 7\ 2\ 1 \Rightarrow \underbrace{111}_7 \ \underbrace{010}_2 \ \underbrace{001}_1 \Rightarrow g(x) = x^8 + x^7 + x^6 + x^4 + 1$

n	M	t	Generator polynomial coefficients
			$g_k \ g_{k-1} \ \dots \ g_1 \ g_0$
7	4	1	13
15	11	1	23
15	7	2	721
15	5	3	2467
31	26	1	45
31	21	2	3551
31	16	3	107657
31	11	5	5423325
31	6	7	313365047
63	57	1	103
63	51	2	12471
63	45	3	1701317
63	39	4	166623567
63	36	5	1033500423
63	30	6	157464165547
63	24	7	17323260404441
63	18	10	1363026512351725
63	16	11	6331141367235453
63	10	13	472622305527250155
63	7	15	52310455435033271737
127	120	1	211
127	113	2	41567
127	106	3	11554743
127	99	4	3447023271
127	92	5	624730022327
127	85	6	1307044763222273

n	M	t	Generator polynomial coefficients
			$g_k \ g_{k-1} \ \dots \ g_1 \ g_0$
127	78	7	26230002166130115
127	71	9	6255010713253127753
127	64	10	1206534025570773100045
127	57	11	335265252505705053517721
127	50	13	54446512523314012421501421
127	43	14	17721772213651227521220574343
127	36	15	314607466522075044764574721735
127	29	21	40311446136767062366753014176155
127	22	23	123376070404722522435445626637647043
127	15	27	22057042445604554770523013762217604353
127	8	31	7047264052751030651476224271567733130217

A12 Table of the Generator Polynomials for RS Codes

Remark

The table lists the generator polynomial coefficients for RS codes of different lengths $n \leq 511$ and different correctable errors (t). The coefficients are given as the decimals associated to $GF(2^k)$ elements and the generator polynomial has the expression:

$$g(x) = (x + \alpha^p)(x + \alpha^{p+1}) \dots (x + \alpha^{p+2t-1}) \text{ or } g(x) = x^{2t} + g_{2t-1}x^{2t-1} + \dots + g_0$$

For example, the generator polynomial for RS(7, 3) code with $p = 1$ is:

$$g(x) = 1 \ 4 \ 1 \ 2 \ 4 \Rightarrow g(x) = x^4 + 4x^3 + x^2 + 2x + 4 \Rightarrow \\ \Rightarrow g(x) = x^4 + \alpha^3x^3 + x^2 + \alpha x + \alpha^3$$

RS(7, 3)	t=2
$p=0$	1 3 6 6 7
$p=1$	1 4 1 2 4
RS(15, 11)	t=2
$p=0$	1 13 5 1 7
$p=1$	1 14 7 4 11
RS(15, 9)	t=3
$p=0$	1 10 13 2 3 5 1
$p=1$	1 11 15 5 7 10 7
RS(15, 7)	t=4
$p=0$	1 14 1 2 14 9 15 5 14
$p=1$	1 15 3 5 3 14 6 12 7

RS(15, 5)	t=5
$p=0$	1 2 2 7 3 10 12 10 14 8 1
$p=1$	1 3 4 10 7 15 3 2 7 2 11
RS(31, 27)	t=2
$p=0$	1 24 18 27 7
$p=1$	1 25 20 30 11
RS(31, 25)	t=3
$p=0$	1 10 8 22 13 20 16
$p=1$	1 11 10 25 17 25 22
RS(31, 23)	t=4
$p=0$	1 3 21 21 16 28 4 24 29
$p=1$	1 4 23 24 20 2 10 31 6
RS(31, 21)	t=5
$p=0$	1 18 30 23 7 5 16 10 26 23 15
$p=1$	1 19 1 26 11 10 22 17 3 1 25
RS(31, 19)	t=6
$p=0$	1 6 21 29 7 29 29 9 29 31 3 30 5
$p=1$	1 7 23 1 11 3 4 16 6 9 13 10 17
RS(31, 17)	t=7
$p=0$	1 27 6 2 14 20 14 18 27 15 22 23 9 12 30
$p=1$	1 28 8 5 18 25 20 25 4 24 1 3 21 25 13
RS(31, 15)	t=8
$p=0$	1 23 12 29 5 30 27 15 18 30 26 13 3 11 9 4 28
$p=1$	1 24 14 1 9 4 2 22 26 8 5 24 15 24 23 19 13
RS(31, 13)	t=9
$p=0$	1 15 10 23 9 24 16 23 29 25 15 26 5 30 1 1 5 27 30
$p=1$	1 16 12 26 13 29 22 30 6 3 25 6 17 12 15 16 21 13 17
RS(31, 11)	t=10
$p=0$	1 22 29 3 26 6 8 5 6 21 14 9 13 31 22 8 16 12 26 7 5
$p=1$	1 23 31 6 30 11 14 12 14 30 24 20 25 13 5 23 1 29 13 26 25
RS(63, 59)	t=2
$p=0$	1 19 40 22 7
$p=1$	1 20 42 25 11
RS(63, 57)	t=3
$p=0$	1 59 47 41 52 6 16
$p=1$	1 60 49 44 56 11 22

RS(63, 55)		t=4															
$p=0$	1	43	58	29	7	36	9	1	29								
$p=1$	1	44	60	32	11	41	15	8	37								
RS(63, 53)		t=5															
$p=0$	1	56	27	45	50	56	59	63	54	29	46						
$p=1$	1	57	29	48	54	61	2	7	62	38	56						
RS(63, 51)		t=6															
$p=0$	1	60	11	42	3	56	23	4	25	12	55	52	4				
$p=1$	1	61	13	45	7	61	29	11	33	21	2	63	16				
RS(63, 49)		t=7															
$p=0$	1	47	8	43	47	50	36	1	49	13	23	32	10	62	29		
$p=1$	1	48	10	46	51	55	42	8	57	22	33	43	22	12	43		
RS(63, 47)		t=8															
$p=0$	1	28	40	62	13	20	49	27	31	42	16	2	10	11	4		
		7	58														
$p=1$	1	29	42	2	17	25	55	34	39	51	26	13	22	24	18		
		22	11														
RS(63, 45)		t=9															
$p=0$	1	22	58	61	63	57	33	15	62	23	16	49	21	62	22		
		37	51	32	28												
$p=1$	1	23	60	1	4	62	39	22	7	32	26	60	33	12	36		
		52	4	49	46												
RS(63, 43)		t=10															
$p=0$	1	54	36	33	59	34	61	30	24	52	25	8	62	24	11		
		3	47	40	62	36	2										
$p=1$	1	55	38	36	63	39	4	37	32	61	35	19	11	37	25		
		18	63	57	17	55	22										
RS(127, 123)		t=2															
$p=0$	1	94	40	97	7												
$p=1$	1	95	42	100	11												
RS(127, 121)		t=3															
$p=0$	1	111	5	124	10	121	16										
$p=1$	1	112	7	127	14	126	22										
RS(127, 119)		t=4															
$p=0$	1	91	33	42	3	49	47	112	29								
$p=1$	1	92	35	45	7	54	53	119	37								
RS(127, 117)		t=5															
$p=0$	1	7	117	106	115	51	124	124	17	43	46						
$p=1$	1	8	119	109	119	56	3	4	25	52	56						
RS(127, 115)		t=6															
$p=0$	1	125	98	3	53	96	90	107	75	36	15	53	67				
$p=1$	1	126	100	6	57	101	96	114	83	45	25	64	79				

RS(255, 237) $t=9$														
$p=0$	1	236	251	109	105	27	166	202	50	129	225	84	90	24
		225	195	254	204	154								
$p=1$	1	217	237	162	99	190	104	126	179	89	198	164	161	11
		194	21	115	114	172								
RS(255,235) $t=10$														
$p=0$	1	36	216	188	10	183	179	197	27	158	5	82	119	38
		142	49	50	46	46	157	191						
$p=1$	1	19	63	171	58	67	170	34	196	212	191	233	238	97
		254	172	181	230	231	208	211						
RS(511, 507) $t=2$														
$p=0$	1	391	41	394	7									
$p=1$	1	392	43	397	11									
RS(511, 505) $t=3$														
$p=0$	1	200	448	39	453	210	16							
$p=1$	1	201	450	42	457	215	22							
RS(511, 503) $t=4$														
$p=0$	1	400	208	119	109	126	222	421	29					
$p=1$	1	401	210	122	113	131	228	428	37					
RS(511, 501) $t=5$														
$p=0$	1	374	119	230	291	117	300	248	146	410	46			
$p=1$	1	375	121	233	295	122	306	255	154	419	56			
RS(511, 499) $t=6$														
$p=0$	1	18	229	314	312	338	81	349	334	347	273	73	67	
$p=1$	1	19	231	317	316	343	87	356	342	356	283	84	79	
RS(511, 497) $t=7$														
$p=0$	1	5	395	124	77	77	268	225	281	103	116	176	460	83
$p=1$	1	6	397	127	81	82	274	232	289	112	126	187	472	96
RS(511, 495) $t=8$														
$p=0$	1	418	285	1	133	288	434	365	358	380	464	333	193	76
		375	12	121										
$p=1$	1	419	287	4	137	293	440	372	366	389	474	344	205	89
		389	27	137										
RS(511, 493) $t=9$														
$p=0$	1	390	472	90	210	352	166	252	200	196	217	286	217	420
		295	192	80	15	154								
$p=1$	1	391	474	93	214	357	172	259	208	205	227	297	229	433
		309	207	96	32	172								
RS(511,491) $t=10$														
$p=0$	1	366	17	118	453	497	299	372	499	139	115	158	26	429
		375	81	56	251	169	26	191						
$p=1$	1	367	19	121	457	502	305	379	507	148	125	169	38	442
		389	96	72	268	187	45	211						

Appendix B: Tables for Information and Entropy Computing

B1 Table for Computing Values of $-\log_2(x)$, $0.01 \leq x \leq 0.99$

x	$-\log_2(x)$	x	$-\log_2(x)$	x	$-\log_2(x)$	x	$-\log_2(x)$
0.00		0.25	2.0000	0.50	1.0000	0.75	0.4150
0.01	6.6439	0.26	1.9434	0.51	0.9714	0.76	0.3959
0.02	5.6439	0.27	1.8890	0.52	0.9434	0.77	0.3771
0.03	5.0589	0.28	1.8365	0.53	0.9159	0.78	0.3585
0.04	4.6439	0.29	1.7859	0.54	0.8890	0.79	0.3401
0.05	4.3219	0.30	1.7370	0.55	0.8625	0.80	0.3219
0.06	4.0589	0.31	1.6897	0.56	0.8365	0.81	0.3040
0.07	3.8365	0.32	1.6439	0.57	0.8110	0.82	0.2863
0.08	3.6439	0.33	1.5995	0.58	0.7859	0.83	0.2688
0.09	3.4739	0.34	1.5564	0.59	0.7612	0.84	0.2515
0.10	3.3219	0.35	1.5146	0.60	0.7370	0.85	0.2345
0.11	3.1844	0.36	1.4739	0.61	0.7131	0.86	0.2176
0.12	3.0589	0.37	1.4344	0.62	0.6897	0.87	0.2009
0.13	2.9434	0.38	1.3959	0.63	0.6666	0.88	0.1844
0.14	2.8365	0.39	1.3585	0.64	0.6439	0.89	0.1681
0.15	2.7370	0.40	1.3219	0.65	0.6215	0.90	0.1520
0.16	2.6439	0.41	1.2863	0.66	0.5995	0.91	0.1361
0.17	2.5564	0.42	1.2515	0.67	0.5778	0.92	0.1203
0.18	2.4739	0.43	1.2176	0.68	0.5564	0.93	0.1047
0.19	2.3959	0.44	1.1844	0.69	0.5353	0.94	0.0893
0.20	2.3219	0.45	1.1520	0.70	0.5146	0.95	0.0740
0.21	2.2515	0.46	1.1203	0.71	0.4941	0.96	0.0589
0.22	2.1844	0.47	1.0893	0.72	0.4739	0.97	0.0439
0.23	2.1203	0.48	1.0589	0.73	0.4540	0.98	0.0291
0.24	2.0589	0.49	1.0291	0.74	0.4344	0.99	0.0145

B2 Table for Computing Values of $-x \cdot \log_2(x)$, $0.001 \leq x \leq 0.999$

x	$-\log_2(x)$	x	$-\log_2(x)$	x	$-\log_2(x)$	x	$-\log_2(x)$
0.000	-	0.250	0.5000	0.500	0.5000	0.750	0.3113
0.001	0.0100	0.251	0.5006	0.501	0.4996	0.751	0.3102
0.002	0.0179	0.252	0.5011	0.502	0.4991	0.752	0.3092
0.003	0.0251	0.253	0.5016	0.503	0.4987	0.753	0.3082
0.004	0.0319	0.254	0.5022	0.504	0.4982	0.754	0.3072
0.005	0.0382	0.255	0.5027	0.505	0.4978	0.755	0.3061
0.006	0.0443	0.256	0.5032	0.506	0.4973	0.756	0.3051
0.007	0.0501	0.257	0.5038	0.507	0.4968	0.757	0.3040
0.008	0.0557	0.258	0.5043	0.508	0.4964	0.758	0.3030
0.009	0.0612	0.259	0.5048	0.509	0.4959	0.759	0.3020
0.010	0.0664	0.260	0.5053	0.510	0.4954	0.760	0.3009
0.011	0.0716	0.261	0.5058	0.511	0.4950	0.761	0.2999
0.012	0.0766	0.262	0.5063	0.512	0.4945	0.762	0.2988
0.013	0.0814	0.263	0.5068	0.513	0.4940	0.763	0.2978
0.014	0.0862	0.264	0.5072	0.514	0.4935	0.764	0.2967
0.015	0.0909	0.265	0.5077	0.515	0.4930	0.765	0.2956
0.016	0.0955	0.266	0.5082	0.516	0.4926	0.766	0.2946
0.017	0.0999	0.267	0.5087	0.517	0.4921	0.767	0.2935
0.018	0.1043	0.268	0.5091	0.518	0.4916	0.768	0.2925
0.019	0.1086	0.269	0.5096	0.519	0.4911	0.769	0.2914
0.020	0.1129	0.270	0.5100	0.520	0.4906	0.770	0.2903
0.021	0.1170	0.271	0.5105	0.521	0.4901	0.771	0.2893
0.022	0.1211	0.272	0.5109	0.522	0.4896	0.772	0.2882
0.023	0.1252	0.273	0.5113	0.523	0.4891	0.773	0.2871
0.024	0.1291	0.274	0.5118	0.524	0.4886	0.774	0.2861
0.025	0.1330	0.275	0.5122	0.525	0.4880	0.775	0.2850
0.026	0.1369	0.276	0.5126	0.526	0.4875	0.776	0.2839
0.027	0.1407	0.277	0.5130	0.527	0.4870	0.777	0.2828
0.028	0.1444	0.278	0.5134	0.528	0.4865	0.778	0.2818
0.029	0.1481	0.279	0.5138	0.529	0.4860	0.779	0.2807
0.030	0.1518	0.280	0.5142	0.530	0.4854	0.780	0.2796
0.031	0.1554	0.281	0.5146	0.531	0.4849	0.781	0.2785
0.032	0.1589	0.282	0.5150	0.532	0.4844	0.782	0.2774
0.033	0.1624	0.283	0.5154	0.533	0.4839	0.783	0.2763
0.034	0.1659	0.284	0.5158	0.534	0.4833	0.784	0.2752
0.035	0.1693	0.285	0.5161	0.535	0.4828	0.785	0.2741
0.036	0.1727	0.286	0.5165	0.536	0.4822	0.786	0.2731
0.037	0.1760	0.287	0.5169	0.537	0.4817	0.787	0.2720

x	$-\log_2(x)$	x	$-\log_2(x)$	x	$-\log_2(x)$	x	$-\log_2(x)$
0.038	0.1793	0.288	0.5172	0.538	0.4811	0.788	0.2709
0.039	0.1825	0.289	0.5176	0.539	0.4806	0.789	0.2698
0.040	0.1858	0.290	0.5179	0.540	0.4800	0.790	0.2687
0.041	0.1889	0.291	0.5182	0.541	0.4795	0.791	0.2676
0.042	0.1921	0.292	0.5186	0.542	0.4789	0.792	0.2665
0.043	0.1952	0.293	0.5189	0.543	0.4784	0.793	0.2653
0.044	0.1983	0.294	0.5192	0.544	0.4778	0.794	0.2642
0.045	0.2013	0.295	0.5196	0.545	0.4772	0.795	0.2631
0.046	0.2043	0.296	0.5199	0.546	0.4767	0.796	0.2620
0.047	0.2073	0.297	0.5202	0.547	0.4761	0.797	0.2609
0.048	0.2103	0.298	0.5205	0.548	0.4755	0.798	0.2598
0.049	0.2132	0.299	0.5208	0.549	0.4750	0.799	0.2587
0.050	0.2161	0.300	0.5211	0.550	0.4744	0.800	0.2575
0.051	0.2190	0.301	0.5214	0.551	0.4738	0.801	0.2564
0.052	0.2218	0.302	0.5217	0.552	0.4732	0.802	0.2553
0.053	0.2246	0.303	0.5220	0.553	0.4726	0.803	0.2542
0.054	0.2274	0.304	0.5222	0.554	0.4720	0.804	0.2530
0.055	0.2301	0.305	0.5225	0.555	0.4714	0.805	0.2519
0.056	0.2329	0.306	0.5228	0.556	0.4708	0.806	0.2508
0.057	0.2356	0.307	0.5230	0.557	0.4702	0.807	0.2497
0.058	0.2383	0.308	0.5233	0.558	0.4696	0.808	0.2485
0.059	0.2409	0.309	0.5235	0.559	0.4690	0.809	0.2474
0.060	0.2435	0.310	0.5238	0.560	0.4684	0.810	0.2462
0.061	0.2461	0.311	0.5240	0.561	0.4678	0.811	0.2451
0.062	0.2487	0.312	0.5243	0.562	0.4672	0.812	0.2440
0.063	0.2513	0.313	0.5245	0.563	0.4666	0.813	0.2428
0.064	0.2538	0.314	0.5247	0.564	0.4660	0.814	0.2417
0.065	0.2563	0.315	0.5250	0.565	0.4654	0.815	0.2405
0.066	0.2588	0.316	0.5252	0.566	0.4648	0.816	0.2394
0.067	0.2613	0.317	0.5254	0.567	0.4641	0.817	0.2382
0.068	0.2637	0.318	0.5256	0.568	0.4635	0.818	0.2371
0.069	0.2662	0.319	0.5258	0.569	0.4629	0.819	0.2359
0.070	0.2686	0.320	0.5260	0.570	0.4623	0.820	0.2348
0.071	0.2709	0.321	0.5262	0.571	0.4616	0.821	0.2336
0.072	0.2733	0.322	0.5264	0.572	0.4610	0.822	0.2325
0.073	0.2756	0.323	0.5266	0.573	0.4603	0.823	0.2313
0.074	0.2780	0.324	0.5268	0.574	0.4597	0.824	0.2301
0.075	0.2803	0.325	0.5270	0.575	0.4591	0.825	0.2290
0.076	0.2826	0.326	0.5272	0.576	0.4584	0.826	0.2278
0.077	0.2848	0.327	0.5273	0.577	0.4578	0.827	0.2266

x	$-\log_2(x)$	x	$-\log_2(x)$	x	$-\log_2(x)$	x	$-\log_2(x)$
0.078	0.2871	0.328	0.5275	0.578	0.4571	0.828	0.2255
0.079	0.2893	0.329	0.5277	0.579	0.4565	0.829	0.2243
0.080	0.2915	0.330	0.5278	0.580	0.4558	0.830	0.2231
0.081	0.2937	0.331	0.5280	0.581	0.4551	0.831	0.2219
0.082	0.2959	0.332	0.5281	0.582	0.4545	0.832	0.2208
0.083	0.2980	0.333	0.5283	0.583	0.4538	0.833	0.2196
0.084	0.3002	0.334	0.5284	0.584	0.4532	0.834	0.2184
0.085	0.3023	0.335	0.5286	0.585	0.4525	0.835	0.2172
0.086	0.3044	0.336	0.5287	0.586	0.4518	0.836	0.2160
0.087	0.3065	0.337	0.5288	0.587	0.4511	0.837	0.2149
0.088	0.3086	0.338	0.5289	0.588	0.4505	0.838	0.2137
0.089	0.3106	0.339	0.5291	0.589	0.4498	0.839	0.2125
0.090	0.3127	0.340	0.5292	0.590	0.4491	0.840	0.2113
0.091	0.3147	0.341	0.5293	0.591	0.4484	0.841	0.2101
0.092	0.3167	0.342	0.5294	0.592	0.4477	0.842	0.2089
0.093	0.3187	0.343	0.5295	0.593	0.4471	0.843	0.2077
0.094	0.3207	0.344	0.5296	0.594	0.4464	0.844	0.2065
0.095	0.3226	0.345	0.5297	0.595	0.4457	0.845	0.2053
0.096	0.3246	0.346	0.5298	0.596	0.4450	0.846	0.2041
0.097	0.3265	0.347	0.5299	0.597	0.4443	0.847	0.2029
0.098	0.3284	0.348	0.5299	0.598	0.4436	0.848	0.2017
0.099	0.3303	0.349	0.5300	0.599	0.4429	0.849	0.2005
0.100	0.3322	0.350	0.5301	0.600	0.4422	0.850	0.1993
0.101	0.3341	0.351	0.5302	0.601	0.4415	0.851	0.1981
0.102	0.3359	0.352	0.5302	0.602	0.4408	0.852	0.1969
0.103	0.3378	0.353	0.5303	0.603	0.4401	0.853	0.1957
0.104	0.3396	0.354	0.5304	0.604	0.4393	0.854	0.1944
0.105	0.3414	0.355	0.5304	0.605	0.4386	0.855	0.1932
0.106	0.3432	0.356	0.5305	0.606	0.4379	0.856	0.1920
0.107	0.3450	0.357	0.5305	0.607	0.4372	0.857	0.1908
0.108	0.3468	0.358	0.5305	0.608	0.4365	0.858	0.1896
0.109	0.3485	0.359	0.5306	0.609	0.4357	0.859	0.1884
0.110	0.3503	0.360	0.5306	0.610	0.4350	0.860	0.1871
0.111	0.3520	0.361	0.5306	0.611	0.4343	0.861	0.1859
0.112	0.3537	0.362	0.5307	0.612	0.4335	0.862	0.1847
0.113	0.3555	0.363	0.5307	0.613	0.4328	0.863	0.1834
0.114	0.3571	0.364	0.5307	0.614	0.4321	0.864	0.1822
0.115	0.3588	0.365	0.5307	0.615	0.4313	0.865	0.1810
0.116	0.3605	0.366	0.5307	0.616	0.4306	0.866	0.1797
0.117	0.3622	0.367	0.5307	0.617	0.4298	0.867	0.1785

x	$-\log_2(x)$	x	$-\log_2(x)$	x	$-\log_2(x)$	x	$-\log_2(x)$
0.118	0.3638	0.368	0.5307	0.618	0.4291	0.868	0.1773
0.119	0.3654	0.369	0.5307	0.619	0.4283	0.869	0.1760
0.120	0.3671	0.370	0.5307	0.620	0.4276	0.870	0.1748
0.121	0.3687	0.371	0.5307	0.621	0.4268	0.871	0.1736
0.122	0.3703	0.372	0.5307	0.622	0.4261	0.872	0.1723
0.123	0.3719	0.373	0.5307	0.623	0.4253	0.873	0.1711
0.124	0.3734	0.374	0.5307	0.624	0.4246	0.874	0.1698
0.125	0.3750	0.375	0.5306	0.625	0.4238	0.875	0.1686
0.126	0.3766	0.376	0.5306	0.626	0.4230	0.876	0.1673
0.127	0.3781	0.377	0.5306	0.627	0.4223	0.877	0.1661
0.128	0.3796	0.378	0.5305	0.628	0.4215	0.878	0.1648
0.129	0.3811	0.379	0.5305	0.629	0.4207	0.879	0.1636
0.130	0.3826	0.380	0.5305	0.630	0.4199	0.880	0.1623
0.131	0.3841	0.381	0.5304	0.631	0.4192	0.881	0.1610
0.132	0.3856	0.382	0.5304	0.632	0.4184	0.882	0.1598
0.133	0.3871	0.383	0.5303	0.633	0.4176	0.883	0.1585
0.134	0.3886	0.384	0.5302	0.634	0.4168	0.884	0.1572
0.135	0.3900	0.385	0.5302	0.635	0.4160	0.885	0.1560
0.136	0.3915	0.386	0.5301	0.636	0.4152	0.886	0.1547
0.137	0.3929	0.387	0.5300	0.637	0.4145	0.887	0.1534
0.138	0.3943	0.388	0.5300	0.638	0.4137	0.888	0.1522
0.139	0.3957	0.389	0.5299	0.639	0.4129	0.889	0.1509
0.140	0.3971	0.390	0.5298	0.640	0.4121	0.890	0.1496
0.141	0.3985	0.391	0.5297	0.641	0.4113	0.891	0.1484
0.142	0.3999	0.392	0.5296	0.642	0.4105	0.892	0.1471
0.143	0.4012	0.393	0.5295	0.643	0.4097	0.893	0.1458
0.144	0.4026	0.394	0.5294	0.644	0.4089	0.894	0.1445
0.145	0.4040	0.395	0.5293	0.645	0.4080	0.895	0.1432
0.146	0.4053	0.396	0.5292	0.646	0.4072	0.896	0.1420
0.147	0.4066	0.397	0.5291	0.647	0.4064	0.897	0.1407
0.148	0.4079	0.398	0.5290	0.648	0.4056	0.898	0.1394
0.149	0.4092	0.399	0.5289	0.649	0.4048	0.899	0.1381
0.150	0.4105	0.400	0.5288	0.650	0.4040	0.900	0.1368
0.151	0.4118	0.401	0.5286	0.651	0.4031	0.901	0.1355
0.152	0.4131	0.402	0.5285	0.652	0.4023	0.902	0.1342
0.153	0.4144	0.403	0.5284	0.653	0.4015	0.903	0.1329
0.154	0.4156	0.404	0.5283	0.654	0.4007	0.904	0.1316
0.155	0.4169	0.405	0.5281	0.655	0.3998	0.905	0.1303
0.156	0.4181	0.406	0.5280	0.656	0.3990	0.906	0.1290
0.157	0.4194	0.407	0.5278	0.657	0.3982	0.907	0.1277

x	$-\text{xlog}_2(\text{x})$	x	$-\text{xlog}_2(\text{x})$	x	$-\text{xlog}_2(\text{x})$	x	$-\text{xlog}_2(\text{x})$
0.158	0.4206	0.408	0.5277	0.658	0.3973	0.908	0.1264
0.159	0.4218	0.409	0.5275	0.659	0.3965	0.909	0.1251
0.160	0.4230	0.410	0.5274	0.660	0.3956	0.910	0.1238
0.161	0.4242	0.411	0.5272	0.661	0.3948	0.911	0.1225
0.162	0.4254	0.412	0.5271	0.662	0.3940	0.912	0.1212
0.163	0.4266	0.413	0.5269	0.663	0.3931	0.913	0.1199
0.164	0.4278	0.414	0.5267	0.664	0.3923	0.914	0.1186
0.165	0.4289	0.415	0.5266	0.665	0.3914	0.915	0.1173
0.166	0.4301	0.416	0.5264	0.666	0.3905	0.916	0.1159
0.167	0.4312	0.417	0.5262	0.667	0.3897	0.917	0.1146
0.168	0.4323	0.418	0.5260	0.668	0.3888	0.918	0.1133
0.169	0.4335	0.419	0.5258	0.669	0.3880	0.919	0.1120
0.170	0.4346	0.420	0.5256	0.670	0.3871	0.920	0.1107
0.171	0.4357	0.421	0.5255	0.671	0.3862	0.921	0.1093
0.172	0.4368	0.422	0.5253	0.672	0.3854	0.922	0.1080
0.173	0.4379	0.423	0.5251	0.673	0.3845	0.923	0.1067
0.174	0.4390	0.424	0.5249	0.674	0.3836	0.924	0.1054
0.175	0.4401	0.425	0.5246	0.675	0.3828	0.925	0.1040
0.176	0.4411	0.426	0.5244	0.676	0.3819	0.926	0.1027
0.177	0.4422	0.427	0.5242	0.677	0.3810	0.927	0.1014
0.178	0.4432	0.428	0.5240	0.678	0.3801	0.928	0.1000
0.179	0.4443	0.429	0.5238	0.679	0.3792	0.929	0.0987
0.180	0.4453	0.430	0.5236	0.680	0.3783	0.930	0.0974
0.181	0.4463	0.431	0.5233	0.681	0.3775	0.931	0.0960
0.182	0.4474	0.432	0.5231	0.682	0.3766	0.932	0.0947
0.183	0.4484	0.433	0.5229	0.683	0.3757	0.933	0.0933
0.184	0.4494	0.434	0.5226	0.684	0.3748	0.934	0.0920
0.185	0.4504	0.435	0.5224	0.685	0.3739	0.935	0.0907
0.186	0.4514	0.436	0.5222	0.686	0.3730	0.936	0.0893
0.187	0.4523	0.437	0.5219	0.687	0.3721	0.937	0.0880
0.188	0.4533	0.438	0.5217	0.688	0.3712	0.938	0.0866
0.189	0.4543	0.439	0.5214	0.689	0.3703	0.939	0.0853
0.190	0.4552	0.440	0.5211	0.690	0.3694	0.940	0.0839
0.191	0.4562	0.441	0.5209	0.691	0.3685	0.941	0.0826
0.192	0.4571	0.442	0.5206	0.692	0.3676	0.942	0.0812
0.193	0.4581	0.443	0.5204	0.693	0.3666	0.943	0.0798
0.194	0.4590	0.444	0.5201	0.694	0.3657	0.944	0.0785
0.195	0.4599	0.445	0.5198	0.695	0.3648	0.945	0.0771
0.196	0.4608	0.446	0.5195	0.696	0.3639	0.946	0.0758
0.197	0.4617	0.447	0.5193	0.697	0.3630	0.947	0.0744

x	$-\log_2(x)$	x	$-\log_2(x)$	x	$-\log_2(x)$	x	$-\log_2(x)$
0.198	0.4626	0.448	0.5190	0.698	0.3621	0.948	0.0730
0.199	0.4635	0.449	0.5187	0.699	0.3611	0.949	0.0717
0.200	0.4644	0.450	0.5184	0.700	0.3602	0.950	0.0703
0.201	0.4653	0.451	0.5181	0.701	0.3593	0.951	0.0689
0.202	0.4661	0.452	0.5178	0.702	0.3583	0.952	0.0676
0.203	0.4670	0.453	0.5175	0.703	0.3574	0.953	0.0662
0.204	0.4678	0.454	0.5172	0.704	0.3565	0.954	0.0648
0.205	0.4687	0.455	0.5169	0.705	0.3555	0.955	0.0634
0.206	0.4695	0.456	0.5166	0.706	0.3546	0.956	0.0621
0.207	0.4704	0.457	0.5163	0.707	0.3537	0.957	0.0607
0.208	0.4712	0.458	0.5160	0.708	0.3527	0.958	0.0593
0.209	0.4720	0.459	0.5157	0.709	0.3518	0.959	0.0579
0.210	0.4728	0.460	0.5153	0.710	0.3508	0.960	0.0565
0.211	0.4736	0.461	0.5150	0.711	0.3499	0.961	0.0552
0.212	0.4744	0.462	0.5147	0.712	0.3489	0.962	0.0538
0.213	0.4752	0.463	0.5144	0.713	0.3480	0.963	0.0524
0.214	0.4760	0.464	0.5140	0.714	0.3470	0.964	0.0510
0.215	0.4768	0.465	0.5137	0.715	0.3460	0.965	0.0496
0.216	0.4776	0.466	0.5133	0.716	0.3451	0.966	0.0482
0.217	0.4783	0.467	0.5130	0.717	0.3441	0.967	0.0468
0.218	0.4791	0.468	0.5127	0.718	0.3432	0.968	0.0454
0.219	0.4798	0.469	0.5123	0.719	0.3422	0.969	0.0440
0.220	0.4806	0.470	0.5120	0.720	0.3412	0.970	0.0426
0.221	0.4813	0.471	0.5116	0.721	0.3403	0.971	0.0412
0.222	0.4820	0.472	0.5112	0.722	0.3393	0.972	0.0398
0.223	0.4828	0.473	0.5109	0.723	0.3383	0.973	0.0384
0.224	0.4835	0.474	0.5105	0.724	0.3373	0.974	0.0370
0.225	0.4842	0.475	0.5102	0.725	0.3364	0.975	0.0356
0.226	0.4849	0.476	0.5098	0.726	0.3354	0.976	0.0342
0.227	0.4856	0.477	0.5094	0.727	0.3344	0.977	0.0328
0.228	0.4863	0.478	0.5090	0.728	0.3334	0.978	0.0314
0.229	0.4870	0.479	0.5087	0.729	0.3324	0.979	0.0300
0.230	0.4877	0.480	0.5083	0.730	0.3314	0.980	0.0286
0.231	0.4883	0.481	0.5079	0.731	0.3305	0.981	0.0271
0.232	0.4890	0.482	0.5075	0.732	0.3295	0.982	0.0257
0.233	0.4897	0.483	0.5071	0.733	0.3285	0.983	0.0243
0.234	0.4903	0.484	0.5067	0.734	0.3275	0.984	0.0229
0.235	0.4910	0.485	0.5063	0.735	0.3265	0.985	0.0215
0.236	0.4916	0.486	0.5059	0.736	0.3255	0.986	0.0201
0.237	0.4923	0.487	0.5055	0.737	0.3245	0.987	0.0186

x	$-\text{xlog}_2(\text{x})$	x	$-\text{xlog}_2(\text{x})$	x	$-\text{xlog}_2(\text{x})$	x	$-\text{xlog}_2(\text{x})$
0.238	0.4929	0.488	0.5051	0.738	0.3235	0.988	0.0172
0.239	0.4935	0.489	0.5047	0.739	0.3225	0.989	0.0158
0.240	0.4941	0.490	0.5043	0.740	0.3215	0.990	0.0144
0.241	0.4947	0.491	0.5039	0.741	0.3204	0.991	0.0129
0.242	0.4954	0.492	0.5034	0.742	0.3194	0.992	0.0115
0.243	0.4960	0.493	0.5030	0.743	0.3184	0.993	0.0101
0.244	0.4966	0.494	0.5026	0.744	0.3174	0.994	0.0086
0.245	0.4971	0.495	0.5022	0.745	0.3164	0.995	0.0072
0.246	0.4977	0.496	0.5017	0.746	0.3154	0.996	0.0058
0.247	0.4983	0.497	0.5013	0.747	0.3144	0.997	0.0043
0.248	0.4989	0.498	0.5009	0.748	0.3133	0.998	0.0029
0.249	0.4994	0.499	0.5004	0.749	0.3123	0.999	0.0014

Appendix C: Signal Detection Elements

C.1 Detection Problem

Signal detection is part of the *statistical decision theory* or hypotheses testing theory. The aim of this processing, made at the receiver, is to decide which was the sent signal, based on the observation of the received signal (observation space). A block- scheme of a system using signal detection is given in Fig C.1.

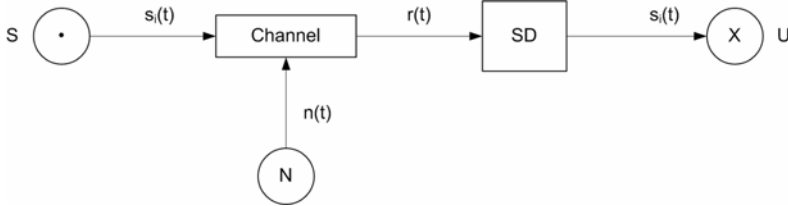


Fig. C.1 Block scheme of a transmission system using signal detection. S- source, N- noise generator, SD- signal detection block, U- user, $s_i(t)$ - transmitted signal, $r(t)$ - received signal, $n(t)$ - noise voltage, $\hat{s}_i(t)$ - estimated signal.

In signal detection block (SD), the received signal $r(t)$ (*observation space*) is observed and, using a *decision criterion*, a decision is made concerning which is the transmitted signal. Decision taken is on thus the affirmation of a hypothesis (H_i). The observation of $r(t)$ can be:

- *discrete observation*: at discrete moments t_i , $i = \overline{1, N}$ samples from $r(t)$ are taken (r_i), the decision being taken on $\vec{r} = (r_1, \dots, r_N)$. If N is variable, the detection is called sequential.
- *continuous observation*: $r(t)$ is observed continuously during the observation time T, and the decision is taken based on $\int_0^T r(t) dt$. It represents the discrete case at limit: $N \rightarrow \infty$.

If the source S is binary, the decision is binary, otherwise M-ary (when the source is M-ary). We will focus only on binary detection, the M-ary case being a generalization of the binary one [1], [4], [6], [7].

The binary source is:

$$S: \begin{pmatrix} s_0(t) & s_1(t) \\ P_0 & P_1 \end{pmatrix}, P_0 + P_1 = 1 \quad (\text{C.1})$$

assumed memoryless, P_0 and P_1 being the a priori probabilities.

Under the assumption of AWGN, the received signal (observation space Δ) is:

$$H_0: r(t) = s_0(t) + n(t) \text{ or } r/s_0 \quad (\text{C.2.a})$$

$$H_1: r(t) = s_1(t) + n(t) \text{ or } r/s_1 \quad (\text{C.2.b})$$

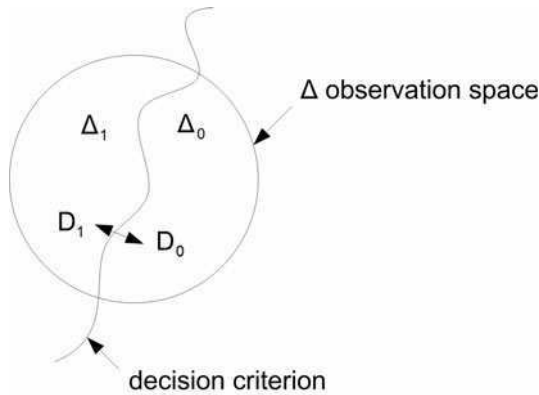


Fig. C.2 Binary decision splits observation space Δ into two disjoint spaces Δ_0 and Δ_1 .

We may have four situations:

- (s_0, D_0) - correct decision in the case of s_0
- (s_1, D_1) - correct decision in the case of s_1
- (s_0, D_1) - wrong decision in the case of s_0
- (s_1, D_0) - wrong decision in the case of s_1

The consequences of these decisions are different and application linked; they can be valued with coefficients named *costs*, C_{ij} : the cost of deciding D_i when s_j was transmitted. For binary decision there are four costs which can be included in cost matrix C :

$$C = \begin{bmatrix} C_{00} & C_{10} \\ C_{01} & C_{11} \end{bmatrix} \quad (\text{C.3})$$

Concerning the costs, always the cost of wrong decisions is higher than those of good decisions (we pay for mistakes):

$$C_{10} \gg C_{00} \text{ and } C_{01} \gg C_{11}$$

In data transmission $C_{00} = C_{11} = 0$ and $C_{01} = C_{10}$ (the consequence of an error on '0' or on '1' is the same).

Then, for binary decision an average cost, named *risk* can be obtained:

$$\begin{aligned} R =: \bar{C} &= \sum_{i=0}^1 \sum_{j=0}^1 C_{ij} P(D_i/s_j) = \\ &= C_{00}P(D_0/s_0) + C_{11}P(D_1/s_1) + C_{01}P(D_0/s_1) + C_{10}P(D_1/s_0) = \\ &= C_{00}P_0P(D_0/s_0) + C_{11}P_1P(D_1/s_1) + C_{01}P_1P(D_0/s_1) + C_{10}P_0P(D_1/s_0) \end{aligned} \quad (C.4)$$

Conditional probabilities $P = (D_i/s_j)$ can be calculated based on conditional *pdfs* (probability density functions): $p(r/s_j)$:

$$P(D_0/s_0) = \int_{\Delta_0} p(r/s_0) dr \quad (C.5.a)$$

$$P(D_1/s_0) = \int_{\Delta_1} p(r/s_0) dr \quad (C.5.b)$$

$$P(D_0/s_1) = \int_{\Delta_0} p(r/s_1) dr \quad (C.5.c)$$

$$P(D_1/s_1) = \int_{\Delta_1} p(r/s_1) dr \quad (C.5.d)$$

Taking into account that the domains Δ_0 and Δ_1 are disjoint, we have:

$$\int_{\Delta_0} p(r/s_0) dr + \int_{\Delta_1} p(r/s_0) dr = 1 \quad (C.6.a)$$

$$\int_{\Delta_0} p(r/s_1) dr + \int_{\Delta_1} p(r/s_1) dr = 1 \quad (C.6.b)$$

Replacing the conditional probabilities $P(D_i/s_j)$ with (C.5.a÷d), and taking into consideration (C.6.a and b), the risk can be expressed only with one domain Δ_0 , or Δ_1 :

$$R = C_{11}P_1 + C_{10}P_0 + \int_{\Delta_0} [p(r/s_1)(C_{01} - C_{11})P_1 - p(r/s_0)(C_{10} - C_{00})] dr \quad (C.4.a)$$

C.2 Signal Detection Criteria

C.2.1 Bayes Criterion

Bayes criterion is the minimum risk criterion and is obtained minimising (C.4.a):

$$\frac{p(r/s_1)}{p(r/s_0)} < \frac{\Delta_0}{\Delta_1} \frac{P_0 C_{10} - C_{00}}{P_1 C_{01} - C_{11}} \tag{C.7}$$

where

$$\frac{p(r/s_1)}{p(r/s_0)} =: \Lambda(r) =: \textit{likelihood ratio} \tag{C.8}$$

$p(r/s_1)$ and $p(r/s_0)$ being known as *likelihood functions* and

$$\frac{P_0 C_{10} - C_{00}}{P_1 C_{01} - C_{11}} = K =: \textit{threshold} \tag{C.9}$$

Then Bayes criterion can be expressed as:

$$\Lambda(r) < K \text{ or } \ln \Lambda(r) < \ln K \tag{C.7.a}$$

and it gives the block scheme of an optimal receiver(Fig. C.3) .

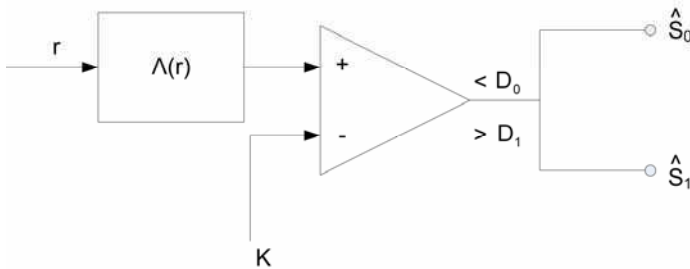


Fig. C.3 Block scheme of an optimal receiver (operating according to Bayes criterion, of minimum risk)

The quality of signal detection processing is appreciated by:

- *Error probability*: P_E (BER)

$$P_E = P_0 P(D_1/s_0) + P_1 P(D_0/s_1) \tag{C.10}$$

Under the assumption of AWGN, the pdf of the noise is $N(0, \sigma_n^2)$:

$$p(n) = \frac{1}{\sqrt{2\pi\sigma_n^2}} e^{-\frac{1}{2\sigma_n^2}n^2} \tag{C.11}$$

and the conditional pdf: $p(r/s_i)$ are also of Gaussian type (Fig. C.4)

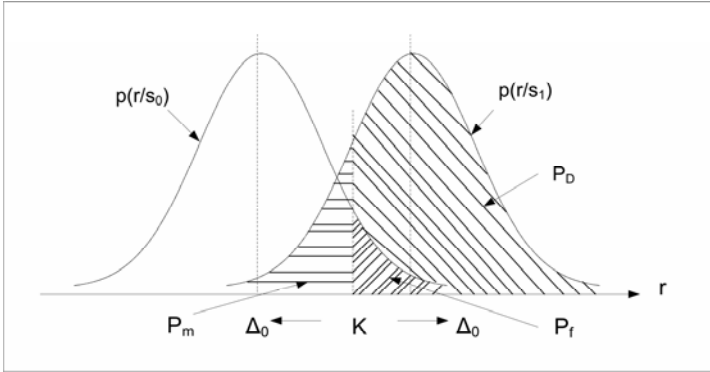


Fig. C.4 Binary detection parameters: P_m - probability of miss, P_D - probability of detection, P_f - probability of false detection

In engineering, the terminology, originating from radar [1] is:

- *probability of false alarm:* P_f

$$P_f = \int_{\Delta_1} p(r/s_0)dr \tag{C.12}$$

- *probability of miss:* P_m

$$P_m = \int_{\Delta_0} p(r/s_1)dr \tag{C.13}$$

- *probability of detection:* P_D

$$P_D = \int_{\Delta_1} p(r/s_1)dr \tag{C.14}$$

- Integrals from normal *pdfs* can be calculated in many ways, one of them being function $Q(y)$, also called *complementary error function (co-error function: erfc)*.

$$Q(y_1) =: \int_{y_1}^{\infty} f(y)dy \tag{C.15}$$

where $f(y)$ is a normal standard pdf, $N(0,1)$:

$$f(y) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}y^2} \quad \text{and} \quad \sigma^2\{y\} = 1 \tag{C.16}$$

with average value $E\{y\} = y = 0$ under the assumption of ergodicity [2]. It's graphical representation is given in Fig. C.5.

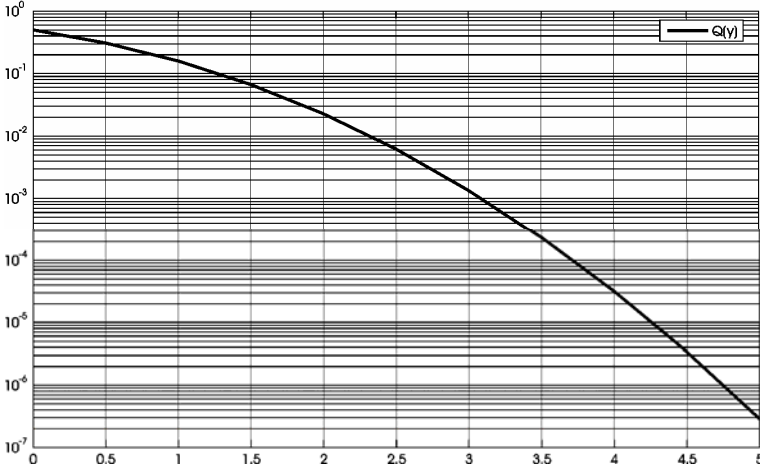


Fig. C.5 Graphical representation of function $Q(y)$

The properties of function $Q(y)$ are:

$$\begin{aligned} Q(-\infty) &= 1 \\ Q(+\infty) &= 0 \\ Q(0) &= \frac{1}{2} \\ Q(-y) &= 1 - Q(y) \end{aligned} \tag{C.17}$$

If the Gaussian pdf is not normal standard, a variable change is used:

$$t = \frac{y - \bar{y}}{\sigma_y}, \quad \text{with } E\{y\} = \bar{y} \neq 0 \quad \text{and} \quad \sigma_n \neq 1 \tag{C.18}$$

C.2.2 Minimum Probability of Error Criterion (Kotelnikov- Siegert)

Under the assumption of:

$$\begin{cases} P_0 = P_1, (P_0 + P_1 = 1) - \text{known} \\ C_{00} = C_{11} = 0 \\ C_{01} = C_{10} = 1 \end{cases} \tag{C.19}$$

The threshold (C.9) becomes:

$$K = \frac{P_0}{P_1}$$

and Bayes risk (C.4), the minimum risk is:

$$R_{\min} = P_1 P_m + P_0 P_f = P_{E\min}$$

from where the name of minimum error probability criteria. Bayes test (C.7) becomes:

$$\Lambda(r) \begin{matrix} D_0 \\ < \frac{P_0}{P_1} \\ > \frac{P_0}{P_1} \\ D_1 \end{matrix} .$$

C.2.3 Maximum a Posteriori Probability Criterion (MAP)

Using Bayes probability relation (2.32), we have:

$$p(rs_i) = p(r)p(s_i/r) = p(s_i)p(r/s_i) \quad (C.21)$$

which gives:

$$\frac{p(r/s_1)P_1}{p(r/s_0)P_0} = \frac{p(s_1/r)}{p(s_0/r)} \begin{matrix} \Delta_0 \\ < 1 \\ > \\ \Delta_1 \end{matrix} \quad (C.22)$$

It can be written as:

$$P_1 p(r/s_1) \begin{matrix} \Delta_0 \\ < \\ > \\ \Delta_1 \end{matrix} P_0 p(r/s_0) \quad (C.22.a)$$

where $p(r/s_0)$ and $p(r/s_1)$ are known as Maximum A Posteriori pdfs.

Remark

(C.22), respectively (C.22.a) are in fact minimum error probability test, showing that MAP for error correction codes is an optimal decoding algorithm: it gives the minimum error probability.

C.2.4 Maximum Likelihood Criterion (R. Fisher)

If to the assumptions (C.19) we add also: $P_0 = P_1$, the threshold (C.9) becomes:

$$K = 1$$

and Bayesian test is:

$$p(r/s_1) \underset{\Delta_1}{\overset{\Delta_0}{>}} p(r/s_0) \quad (\text{C.23})$$

Remark

The assumptions $\begin{cases} C_{00} = C_{11} = 0 \\ C_{01} = C_{10} = 1 \\ P_0 = P_1 = \frac{1}{2} \end{cases}$ are basically those from data processing, this is

why maximum likelihood criterion ($K = 1$) is the used decision criterion in data processing.

C.3 Signal Detection in Data Processing ($K = 1$)

C.3.1 Discrete Detection of a Unipolar Signal

Hypotheses:

- unipolar signal (in baseband): $\begin{cases} s_0(t) = 0 \\ s_1(t) = A = ct \end{cases}$
- AWGN: $N(0, \sigma_n^2)$; $r(t) = s_i(t) + n(t)$
- $T =$ bit duration = observation time
- Discrete observation with N samples per observation time (T) $\Rightarrow \bar{r} = (r_1, \dots, r_N)$
- $C_{00} = C_{11} = 0$, $C_{01} = C_{10} = 1$, P_0, P_1 , with $P_0 = P_1 = \frac{1}{2}$

a. Likelihood ratio calculation

$$H_0 : r(t) = s_0(t) + n(t) = n(t) \rightarrow r/s_0 = n(t)$$

A sample $r_i = n_i \in N(0, \sigma_n^2)$ and the N samples are giving $\bar{r} = (r_1, \dots, r_N) = (n_1, \dots, n_N)$

$$p(r_i/s_0) = \frac{1}{\sqrt{2\pi\sigma_n^2}} e^{-\frac{1}{2\sigma_n^2}r_i^2}$$

$$p(\bar{r}/s_0) = \left[\frac{1}{\sqrt{2\pi\sigma_n^2}} \right]^N e^{-\frac{1}{2\sigma_n^2}\sum_{i=1}^N r_i^2}$$

$$H_1 : r(t) = s_1(t) + n(t) = A + n(t)$$

$$r_i = A + n_i \in N(A, \sigma_n^2) \Rightarrow p(r_i/s_1) = \frac{1}{\sqrt{2\pi\sigma_n^2}} e^{-\frac{1}{2\sigma_n^2}(r_i-A)^2}$$

$$p(\vec{r}/s_1) = \left[\frac{1}{\sqrt{2\pi\sigma_n^2}} \right]^N e^{-\frac{1}{2\sigma_n^2} \sum_{i=1}^N (r_i-A)^2}$$

$$\Lambda(r) = e^{\frac{A}{\sigma_n^2} \sum_{i=1}^N r_i - \frac{A^2 N}{2\sigma_n^2}}$$

b. Minimum error probability test, applied to the logarithmic relation:

$$\ln \Lambda(\vec{r}) \underset{\Delta_1}{\overset{\Delta_0}{>}} \ln K$$

$$\frac{A}{\sigma_n^2} \sum_{i=1}^N r_i - \frac{A^2 N}{2\sigma_n^2} \underset{\Delta_1}{\overset{\Delta_0}{>}} \ln K, \text{ or}$$

$$\sum_{i=1}^N r_i \underset{\Delta_1}{\overset{\Delta_0}{>}} \frac{\sigma_n^2}{A} \ln K + \frac{AN}{2} \tag{C.24}$$

where $\sum_{i=1}^N r_i$ represents a sufficient statistics, meaning that it is sufficient to take the decisions and

$$\frac{\sigma_n^2}{A} \ln K + \frac{AN}{2} = K' \tag{C.25}$$

represents a threshold depending on: the power of the noise on the channel (σ_n^2), the level of the signal (A), the number of samples (N) and P_0, P_1 (through $K = P_0/P_1$).

Relation (C.24) direction to the block-scheme of an optimal receiver:

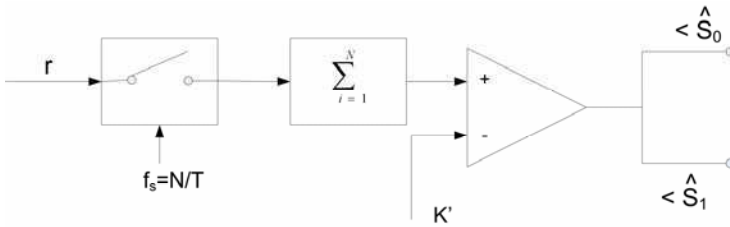


Fig. C.6 Block-scheme of the optimal receiver for unipolar signal and discrete observation.

Remark

If $K=1$ and $N=1$ (one sample per bit, taken at $T/2$) and $P_0 = P_1$ ($K=1$), the decision relation (C.24) becomes:

$$\begin{aligned} &\Delta_0 \\ &< \frac{A}{2} \\ &r_i > \frac{A}{2} \\ &\Delta_0 \end{aligned} \tag{C.24.a}$$

c. Error probability of the optimal receiver variable is

According to (C.24), the decision variable is $\sum_{i=1}^N r_i \in n(E[y], \sigma^2)$, $E[y]$ being the variable and σ^2 the dispersion. Making the variable change

$$y = \frac{\sum_{i=1}^N r_i}{\sigma_n \sqrt{N}} \tag{C.26}$$

a normalization is obtained: $\sigma^2[y] = 1$.

Using this new variable, the decision relation becomes:

$$\frac{\sum_{i=1}^N r_i}{\sigma_n \sqrt{N}} > \frac{\Delta_0}{A \sqrt{N}} > \frac{\sigma_n}{A \sqrt{N}} \ln K + \frac{A \sqrt{N}}{2 \sigma_n} \tag{C.24.b}$$

If we note:

$$\frac{\sigma_n}{A \sqrt{N}} \ln K + \frac{A \sqrt{N}}{2 \sigma_n} = \mu \tag{C.27}$$

the decision relation (C.24.b) is:

$$\begin{aligned} & \Delta_0 \\ & y < \mu \\ & > \mu \\ & \Delta_1 \end{aligned} \tag{C.24.c}$$

Under the two assumptions H_0 and H_1 , the pdfs of y are:

$$p(y/s_0) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}y^2} \tag{C.28.a}$$

$$p(y/s_1) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(y - \frac{A\sqrt{N}}{\sigma_n})^2} \tag{C.28.b}$$

graphically represented in Fig. C.7.

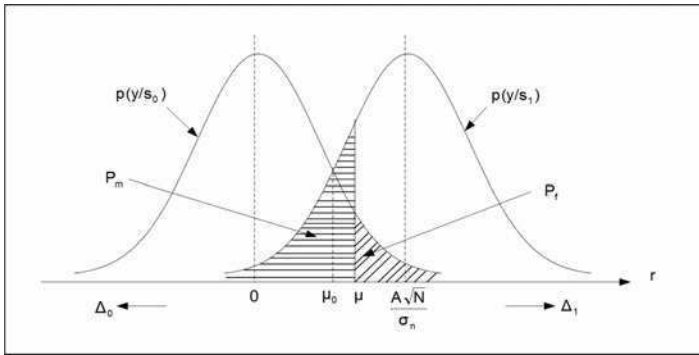


Fig. C.7 Graphical representation of pdfs of decision variables for unipolar decision and discrete observation.

$$P_f = \int_{\mu}^{\infty} p(y/s_0) dy = Q(\mu) \tag{C.29}$$

$$P_m = \int_{-\infty}^{\mu} p(y/s_1) dy = 1 - Q(\mu - \frac{A\sqrt{N}}{\sigma_n}) \tag{C.30}$$

It is a particular value μ_0 for which $P_f = P_m$:

$$Q(\mu_0) = 1 - Q(\mu_0 - \frac{A\sqrt{N}}{\sigma_n})$$

It follows that:

$$\mu_0 = \frac{1}{2} \frac{A\sqrt{N}}{\sigma_n} \quad (\text{C.31})$$

and according to (C.27), μ_0 is obtained if $K=1$, which means that $P_0 = P_1 = \frac{1}{2}$ and:

$$P_E = Q\left(\frac{1}{2} \frac{A\sqrt{N}}{\sigma_n}\right) \quad (\text{C.32})$$

or

$$P_E = Q\left(\sqrt{\frac{1}{2}\xi N}\right) \quad (\text{C.33})$$

where ξ designates the SNR:

$$\text{SNR} = \xi = \frac{P_s}{P_n} = \frac{\frac{A^2}{2R}}{\frac{\sigma_n^2}{R}} = \frac{1}{2} \frac{A^2}{\sigma_n^2} \quad (\text{C.34})$$

It follows that the minimum required SNR for $P_E = 10^{-5}$ for $N=1$ (the required value in PCM systems, see 3.3.5) is approximately 15dB (15,6 dB), which is the threshold value of the required input SNR: ξ_{s10} which separate the regions of decision noise to that one of quantisation noise- Fig. 3.8.

C.3.2 Discrete Detection of Polar Signal

Hypotheses:

- Polar signal in baseband: $s_0(t) = B, s_1(t) = A, B < A, B = -A$
- AWGN: $N(0, \sigma_n^2)$
- T- observation time = bit duration
- Discrete observation with N samples per T
- $C_{00} = C_{11} = 0, C_{01} = C_{10} = 1$

Following the steps similar to those from C.3.1, we obtain:

a.

$$\Lambda(\vec{r}) = e^{-\frac{1}{2\sigma_n^2} \left[\sum_{i=1}^N (r_i - A)^2 - \sum_{i=1}^N (r_i - B)^2 \right]} \quad (\text{C.35})$$

$$\text{b. } \ln \Lambda(\hat{r}) \underset{\Delta_1}{\overset{\Delta_0}{>}} \ln K$$

$$\sum_{i=1}^N r_i \underset{\Delta_1}{\overset{\Delta_0}{>}} \frac{\sigma_n^2}{A-B} \ln K + \frac{N(A+B)}{2} \quad (\text{C.36})$$

For polar signal: $B = -A$, and $K=1$, the threshold of the comparator is $K' = 0$; if $N=1$, the comparator will decide '1' for positive samples and '0' for negative ones.

c. In order to calculate the quality parameters: P_E , a variable change for normalization of the decision variable is done:

$$\frac{\sum_{i=1}^N r_i}{\sigma_n \sqrt{N}} \underset{\Delta_1}{\overset{\Delta_0}{>}} \underbrace{\frac{\sigma_n}{(A-B)\sqrt{N}} \ln K + \frac{(A+B)\sqrt{N}}{2\sigma_n}}_{\mu} \quad (\text{C.37})$$

The decision variable pdf under the two hypotheses is:

$$p(y/s_0) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2} \left(y - \frac{B\sqrt{N}}{\sigma_n} \right)^2} \quad (\text{C.38.a})$$

$$p(y/s_1) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2} \left(y - \frac{A\sqrt{N}}{\sigma_n} \right)^2} \quad (\text{C.38.b})$$

The threshold μ_0 for which $P_f = P_m$ is:

$$\mu_0 = \frac{(A+B)\sqrt{N}}{2\sigma_n} \quad (\text{C.39})$$

which implies $K=1$ ($P_0 = P_1 = \frac{1}{2}$).

If $B = -A$, the polar case,

$$P_E = Q\left(\frac{A\sqrt{N}}{\sigma_n}\right) = Q(\sqrt{\xi N}) \quad (\text{C.40})$$

Compared with the unipolar case, relation (C.33), we may notice that the same BER (P_E) is obtained in the polar case with 3dB less SNR.

C.3.3 Continuous Detection of Known Signal

Hypotheses:

- $\begin{cases} s_0(t) = 0 \\ s_1(t) = s(t), \text{ of finite energy } E = \int_0^T s^2(t) dt \end{cases}$
- T- observation time
- continuous observation: $\bar{r} = (r_1, \dots, r_{N \rightarrow \infty})$
- AWGN: $n(t) \in N(0, \sigma_n^2)$, $r(t) = s_i(t) + n(t)$

a. Calculation of $\Lambda(r) = \frac{p(r/s_1)}{p(r/s_0)}$

Continuous observation means $N \rightarrow \infty$. We shall express the received signal $r(t)$ as a series of orthogonal functions $v_i(t)$ (Karhunen-Loeve expansion [2]) in such a way that the decision could be taken using only one function (coordinate), meaning that it represents the sufficient statistics.

$$r(t) = \lim_{N \rightarrow \infty} \sum_{i=1}^N r_i v_i(t) \quad (C.41)$$

The functions $v_i(t)$ are chosen to represent an orthonormal (orthogonal and normalised) system.

$$\int_0^T v_i(t) v_j(t) dt = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases} \quad (C.42)$$

The coefficients r_i are given by:

$$r_i = \int_0^T r(t) v_i(t) dt \quad (C.43)$$

and represent the coordinates of $r(t)$ on the observation interval $[0, T]$. In order to have $v_1(t)$ as sufficient statistics, we chose:

$$v_1(t) = \frac{s(t)}{\sqrt{E}} \quad (C.44)$$

and r_1 is:

$$r_1 = \int_0^T r(t) v_1(t) dt = \frac{1}{\sqrt{E}} \int_0^T r(t) s(t) dt \quad (C.45)$$

We show that higher order coefficients: r_i with $i > 1$, do not affect the likelihood ratio:

$$\Lambda(\bar{r}) = \frac{p(\bar{r}/s_1)}{p(\bar{r}/s_0)} = \frac{\prod_{i=1}^{N \rightarrow \infty} p(r_i/s_1)}{\prod_{i=1}^{N \rightarrow \infty} p(r_i/s_0)} = \frac{p(r_1/s_1)}{p(r_1/s_0)} \quad (\text{C.46})$$

the contribution of higher order coefficients being equal in the likelihood ratio:

$$\begin{aligned} r_i/s_0 &= \int_0^T n(t)v_i(t)dt \\ r_i/s_1 &= \int_0^T [s(t) + n(t)]v_i(t)dt = \int_0^T s(t)v_i(t)dt + \int_0^T n(t)v_i(t)dt = \\ &= \int_0^T n(t)v_i(t)dt = r_i/s_0 \end{aligned} \quad (\text{C.47})$$

because $\int_0^T s(t)v_i(t)dt = \sqrt{E} \int_0^T v_1(t)v_i(t)dt = 0$, based on the orthogonality of $v_1(t)$ and $v_i(t)$.

Then, $v_1(t) = s(t)/\sqrt{E}$ is a sufficient statistics.

$$\Lambda(\bar{r}) = \frac{p(r_1/s_1)}{p(r_1/s_0)}$$

$$H_0 : r(t) = s_0(t) + n(t) = n(t)$$

$$r_1/s_0 = \int_0^T n(t)v_1(t)dt = \frac{1}{\sqrt{E}} \int_0^T n(t)s(t)dt = \gamma \quad (\text{C.48})$$

and has a normal pdf.

The average value of r_1/s_0 : $\bar{r}_1/s_0 = 0$ based on $n(t) \in N(0, \sigma_n^2)$ and $\sigma^2[r_1/s_0] = \sigma^2 \left[\frac{1}{\sqrt{E}} \int_0^T n(t)s(t)dt \right] = \frac{1}{E} \sigma_n^2 T E = \sigma_n^2 T$

It follows that:

$$p(r_1/s_0) = \frac{1}{\sqrt{2\pi\sigma_n^2 T}} e^{-\frac{1}{2\sigma_n^2 T} r_1^2} \quad (\text{C.49})$$

$$H_1 : r(t) = s_1(t) + n(t) = s(t) + n(t)$$

$$\begin{aligned} r_1/s_1 &= \frac{1}{\sqrt{E}} \int_0^T [s(t) + n(t)]s(t)dt = \\ &= \frac{1}{\sqrt{E}} \int_0^T s^2(t)dt + \frac{1}{\sqrt{E}} \int_0^T n(t)s(t) = \sqrt{E} + \gamma \end{aligned} \quad (C.50)$$

$$p(r_1/s_1) = \frac{1}{\sqrt{2\pi\sigma_n^2 T}} e^{-\frac{1}{2\sigma_n^2 T} (r_1 - \sqrt{E})^2} \quad (C.51)$$

$$\Lambda(\bar{r}) = e^{-\frac{1}{2\sigma_n^2 T} (r_1^2 - 2\sqrt{E}r_1 + E - r_1^2)}$$

b. The decision criterion is:

$$\ln\Lambda(\bar{r}) \underset{\Delta_1}{\overset{\Delta_0}{>}} \ln K$$

$$r_1 \underset{\Delta_1}{\overset{\Delta_0}{>}} \frac{\sigma_n^2 T}{\sqrt{E}} \ln K + \frac{\sqrt{E}}{2} \quad (C.52)$$

If r_1 is replaced with (C.45), the decision relation becomes:

$$\int_0^T r(t)s(t)dt \underset{\Delta_1}{\overset{\Delta_0}{>}} \sigma_n^2 T \ln K + \frac{E}{2} \quad (C.53)$$

where $\sigma_n^2 T \ln K + \frac{E}{2} = K'$.

The block- scheme of the optimal receiver can be implemented in two ways: correlator-base (Fig. C.8.a), or matched filter-based (Fig. C.8.b).

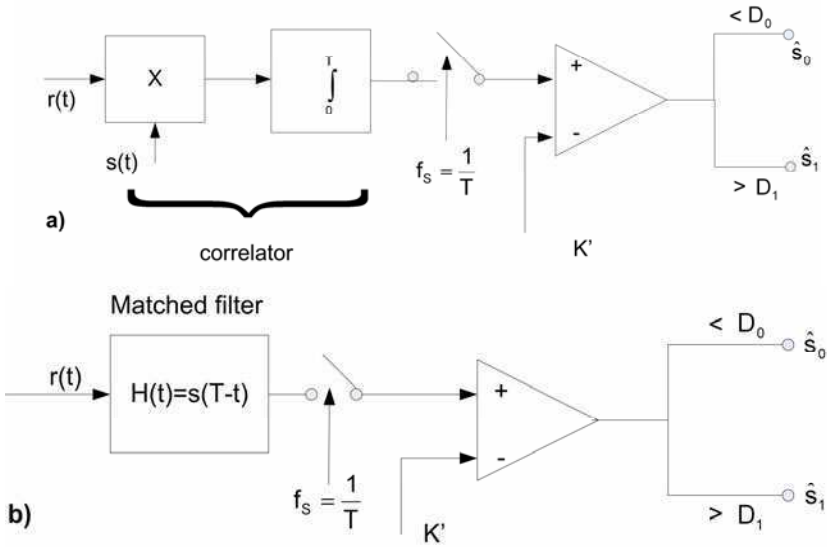


Fig. C.8 Block Scheme of an optimal receiver with continuous observation decision for one known signal $s(t)$: a) correlator based implementation; b) matched filter implementation

c. Decision relation is (C.52). Making a variable change to obtain unitary dispersion, we get:

$$\frac{r_1}{\sqrt{\sigma_n^2 T}} \underset{\Delta_1}{>} \underset{\Delta_0}{<} \sqrt{\frac{\sigma_n^2 T}{E}} \ln K + \frac{1}{2} \sqrt{\frac{E}{\sigma_n^2 T}} \tag{C.54}$$

Using the notations:

$$z = \frac{r_1}{\sqrt{\sigma_n^2 T}} \tag{C.55}$$

$$\mu = \sqrt{\frac{\sigma_n^2 T}{E}} \ln K + \frac{1}{2} \sqrt{\frac{E}{\sigma_n^2 T}} \tag{C.56}$$

the pdfs of the new variable z are:

$$p(z/s_0) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2} \tag{C.57}$$

$$p(z/s_1) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(z - \sqrt{\frac{E}{\sigma_n^2 T}})^2} \tag{C.58}$$

which are represented in Fig. C.9.

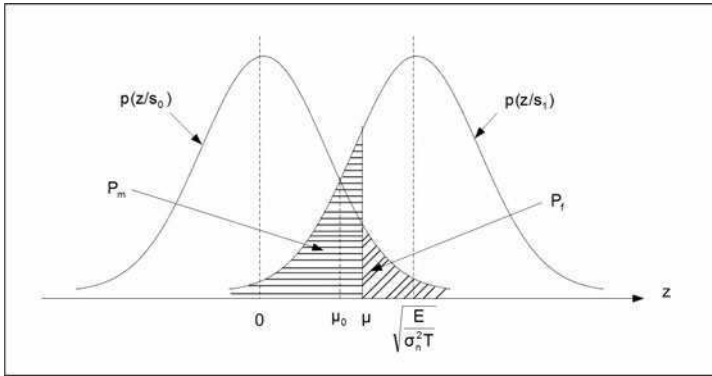


Fig. C.9 Graphical representation of $p(z/s_0)$ and $p(z/s_1)$.

The probabilities occurring after decision are:

$$P(D_0/s_0) = \int_{-\infty}^{\mu} p(z/s_0) dz = 1 - Q(\mu) \quad (\text{C.59.a})$$

$$P(D_1/s_1) = P_D = \int_{\mu}^{\infty} p(z/s_1) dz = Q\left(\mu - \sqrt{\frac{E}{\sigma_n^2 T}}\right) \quad (\text{C.59.b})$$

$$P(D_0/s_1) = P_m = \int_{-\infty}^{\mu} p(z/s_1) dz = 1 - Q\left(\mu - \sqrt{\frac{E}{\sigma_n^2 T}}\right) \quad (\text{C.59.c})$$

$$P(D_1/s_0) = P_f = \int_{\mu}^{\infty} p(z/s_0) dz = Q(\mu) \quad (\text{C.59.d})$$

The particular value μ_0 for which $P_f = P_m$ is:

$$\mu_0 = \frac{1}{2} \sqrt{\frac{E}{\sigma_n^2 T}} \quad (\text{C.60})$$

and according to (C.56) is obtained for $K=1$.

In this case, $K=1$, the bit error rate is:

$$P_E = Q\left(\frac{1}{2} \sqrt{\frac{E}{\sigma_n^2 T}}\right) \quad (\text{C.61})$$

which can be expressed also as a function of the ratio E_b/N_0

$$\begin{cases} E_b = \frac{E}{2} \\ \sigma_n^2 T = N_0 B T = N_0 \frac{1}{2T} T = \frac{N_0}{2} \end{cases}$$

$$P_E = Q\left(\sqrt{\frac{E}{2} \cdot \frac{1}{2} \cdot \frac{2}{N_0}}\right) = Q\left(\sqrt{\frac{E_b}{N_0}}\right) \quad (\text{C.62})$$

It follows that the required E_b/N_0 for 10^{-5} BER is 12.6dB, with 3dB less than the required ξ in the discrete observation with 1 sample per bit.

C.3.4 Continuous Detection of Two Known Signals

Hypotheses:

- $\begin{cases} s_0(t) \text{ of finite energy } E_0 = \int_0^T s_0^2(t) dt \\ s_1(t) \text{ of finite energy } E_1 = \int_0^T s_1^2(t) dt \end{cases}$
 - T- observation time
 - continuous observation : $\vec{r} = (r_1, \dots, r_{N \rightarrow \infty})$
 - AWGN: $n(t) \in N(0, \sigma_n^2)$, $r(t) = s_i(t) + n(t)$
- a.

$$\Lambda(\vec{r}) = \frac{p(\vec{r}/s_1)}{p(\vec{r}/s_0)}$$

$$r(t) = \lim_{N \rightarrow \infty} \sum_{i=1}^N r_i v_i(t)$$

If the first two functions: $v_1(t)$ and $v_2(t)$ are properly chosen, $\Lambda(\vec{r})$ can be expressed only by the coordinates r_1 and r_2 , which represent the sufficient statistics.

$$v_1(t) = \frac{1}{\sqrt{E_1}} s_1(t) \quad (\text{C.63})$$

$$v_2(t) = \frac{1}{\sqrt{1-\rho^2}} \left[\frac{s_0(t)}{\sqrt{E_0}} - \rho \frac{s_1(t)}{\sqrt{E_1}} \right] \quad (\text{C.64})$$

where ρ is the *correlation coefficient*.

$$\rho = \frac{1}{\sqrt{E_0 E_1}} \int_0^T s_0(t) s_1(t) dt \quad (\text{C.65})$$

Easily can be checked that:

$$\int_0^T v_1^2(t) dt = 1 \quad (\text{C.66.a})$$

$$\int_0^T v_2^2(t) dt = 1 \quad (\text{C.66.b})$$

$$\int_0^T v_1(t)v_2(t) dt = 0 \quad (\text{C.66.c})$$

Higher order functions $v_i(t)$, with $i > 2$, if orthogonal can be any; it means that the coordinates r_i , with $i > 2$ do not depend on the hypotheses H_0 , H_1 and then r_1 and r_2 are the sufficient statistics.

$$H_0 : r_i/s_0 = \int_0^T [s_0(t) + n(t)]v_i(t) dt = \int_0^T n(t)v_i(t) dt \quad (\text{C.67.a})$$

$$H_1 : r_i/s_1 = \int_0^T [s_1(t) + n(t)]v_i(t) dt = \int_0^T n(t)v_i(t) dt \quad (\text{C.67.b})$$

Consequently, the likelihood ratio is:

$$\Lambda(\vec{r}) = \frac{p(r_1/s_1)p(r_2/s_1)}{p(r_1/s_0)p(r_2/s_0)} \quad (\text{C.68})$$

The coordinates r_1 and r_2 are:

$$r_1 = \frac{1}{\sqrt{E_1}} \int_0^T r(t)s_1(t) dt \quad (\text{C.69})$$

$$r_2(t) = \frac{1}{\sqrt{1-\rho^2}} \left[\frac{1}{\sqrt{E_0}} \int_0^T r(t)s_0(t) dt - \frac{\rho}{\sqrt{E_1}} \int_0^T r(t)s_1(t) dt \right] \quad (\text{C.70})$$

Under the two hypotheses, we have:

$$r_1/s_0 = \frac{1}{\sqrt{E_1}} \int_0^T [s_0(t) + n(t)]s_1(t) dt = \sqrt{E_0}\rho + \rho_1 \quad (\text{C.71})$$

with

$$\rho_1 = \frac{1}{\sqrt{E_1}} \int_0^T n(t)s_1(t) dt \quad (\text{C.72})$$

$$r_1/s_1 = \frac{1}{\sqrt{E_1}} \int_0^T [s_1(t) + n(t)]s_1(t)dt = \sqrt{E_1} + \rho_1 \tag{C.73}$$

$$r_2/s_0 = \frac{1}{\sqrt{1-\rho^2}} \left\{ \frac{1}{\sqrt{E_0}} \int_0^T [s_0(t) + n(t)]s_0(t)dt - \frac{\rho}{\sqrt{E_1}} \int_0^T [s_0(t) + n(t)]s_1(t)dt \right\} =$$

$$= \sqrt{E_0} \sqrt{1-\rho^2} + \frac{\rho_0 - \rho\rho_1}{\sqrt{1-\rho^2}} \tag{C.74}$$

where

$$\rho_0 = \frac{1}{\sqrt{E_0}} \int_0^T n(t)s_0(t)dt \tag{C.75}$$

$$r_2/s_1 = \frac{1}{\sqrt{1-\rho^2}} \left\{ \frac{1}{\sqrt{E_0}} \int_0^T [s_1(t) + n(t)]s_0(t)dt - \frac{\rho}{\sqrt{E_1}} \int_0^T [s_1(t) + n(t)]s_1(t)dt \right\} = \frac{\rho_0 - \rho\rho_1}{\sqrt{1-\rho^2}} \tag{C.76}$$

Under the assumption of noise absence: $n(t)=0$, $(r_1/s_0 = \sqrt{E_0}\rho$, $r_2/s_0 = \sqrt{E_0}\sqrt{1-\rho^2})$ are the coordinate of point M_0 , and $(r_1/s_1 = \sqrt{E_1}$, $r_2/s_1 = 0)$ the coordinate of point M_1 , represented in the space (r_1, r_2) , Fig. C.10.

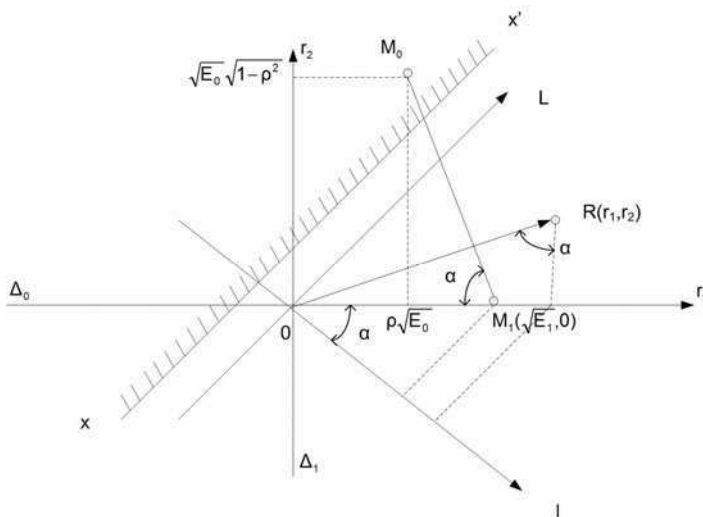


Fig. C.10 Observation space in dimensions (r_1, r_2) .

The separation line between Δ_0 and Δ_1 is the line $(x x')$ orthogonal on M_0M_1 . If we rotate the coordinates such that l be parallel with M_0M_1 , the information necessary to take the decision is contained only in the coordinate l which plays the role of sufficient statistics. Assume that the received vector is the point $R (r_1, r_2)$.

$$l = r_1 \cos \alpha - r_2 \sin \alpha \quad (C.77)$$

which is a Gaussian, based on the Gaussianity of r_1 and r_2 .

$$\begin{aligned} \cos \alpha &= \frac{\sqrt{E_1} - \rho \sqrt{E_0}}{\sqrt{(\sqrt{E_0} \sqrt{1-\rho^2})^2 + (\sqrt{E_1} - \rho \sqrt{E_0})^2}} = \\ &= \frac{\sqrt{E_1} - \rho \sqrt{E_0}}{\sqrt{E_1 - 2\rho \sqrt{E_0 E_1} + E_0}} \end{aligned} \quad (C.78)$$

$$\sin \alpha = \frac{\sqrt{E_0} \sqrt{1-\rho^2}}{\sqrt{E_1 - 2\rho \sqrt{E_0 E_1} + E_0}} \quad (C.79)$$

If we introduce the notation:

$$z = \frac{1}{\sqrt{\sigma_n^2 T}} \quad (C.80)$$

and l is replaced with (C.77), (C.78) and (C.79), we obtain:

$$z = \frac{1}{\sqrt{\sigma_n^2 T} \sqrt{E_0 - 2\rho \sqrt{E_0 E_1} + E_1}} \left[\int_0^T r(t) s_1(t) dt - \int_0^T r(t) s_0(t) dt \right] \quad (C.81)$$

The likelihood ratio can be written as a function of z , which plays the role of sufficient statistics.

$$\Lambda(z) = \frac{\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2} \left(z - \frac{a_1}{\sqrt{\sigma_n^2 T}} \right)^2}}{\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2} \left(z - \frac{a_0}{\sqrt{\sigma_n^2 T}} \right)^2}} \quad (C.82)$$

with

$$a_1 = \bar{l}/s_1 = \frac{E_1 - \rho \sqrt{E_0 E_1}}{\sqrt{E_1 - 2\rho \sqrt{E_0 E_1} + E_0}} \quad (C.83)$$

$$a_0 = \bar{l}/s_0 = -\frac{E_0 - \rho\sqrt{E_0E_1}}{\sqrt{E_1 - 2\rho\sqrt{E_0E_1} + E_0}} \tag{C.84}$$

b. The decision criterion, applied to the logarithmic relation, gives

$$z \underset{\Delta_1}{<} \frac{\sqrt{\sigma_n^2 T}}{a_1 - a_0} \ln K + \frac{1}{2} \frac{a_1 + a_0}{\sqrt{\sigma_n^2 T}} \tag{C.85}$$

If we note:

$$\mu = \frac{\sqrt{\sigma_n^2 T}}{a_1 - a_0} \ln K + \frac{1}{2} \frac{a_1 + a_0}{\sqrt{\sigma_n^2 T}} \tag{C.86}$$

The decision relation is:

$$z \underset{>}{<} \mu \tag{C.85.a}$$

which can be written, based on (C.81):

$$\int_0^T r(t)s_1(t)dt - \int_0^T r(t)s_0(t)dt \underset{\Delta_1}{<} \frac{\Delta_0}{\sigma_n^2 T} \ln K + \frac{E_1 - E_0}{2} \tag{C.87}$$

and represents the decision relation.

$$\sigma_n^2 T \ln K + \frac{E_1 - E_0}{2} = K' \tag{C.88}$$

represents the threshold of the comparator.

The implementation of the decision relation (C.87) gives the block - scheme of the optimal receiver (Fig. C.11).

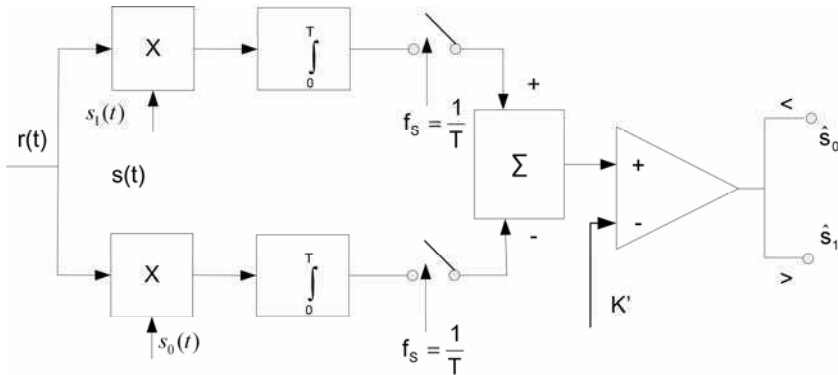


Fig. C.11 Block- scheme of an optimal receiver for continuous decision with two known signal (correlator based implementation).

As presented in Fig. C.8, the correlator can be replaced with matched filters (Fig. C.8.b)

c. The decision variable z , under the two hypotheses is represented in Fig. C.12

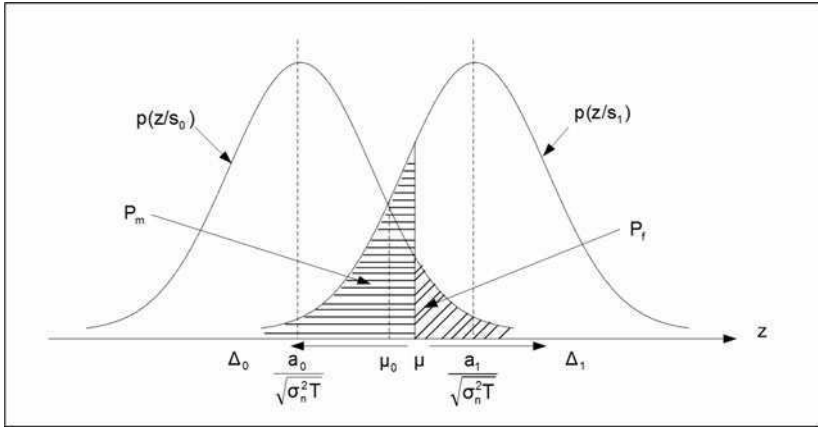


Fig. C.12 Representation of the decision process in continuous detection of two known signals

The distance between the maxima of $p(z/s_0)$ and $p(z/s_1)$ is:

$$\gamma = \frac{a_1}{\sqrt{\sigma_n^2 T}} - \frac{a_0}{\sqrt{\sigma_n^2 T}} = \frac{\sqrt{E_1 - 2\rho\sqrt{E_0 E_1} + E_0}}{\sqrt{\sigma_n^2 T}} \tag{C.89}$$

P_f and P_m are decreasing, which means P_E is decreasing, if γ is greater. The greatest γ , for $E_0 + E_1 = ct$, is obtained when $\rho = -1$ and $E_0 + E_1 = E$, respectively when the performance of the optimal receiver depends only on its energy.

$$s_0(t) = -s_1(t) \tag{C.90}$$

We can notice that the shape of the signal has no importance, the performance of the optimal receiver depending on its energy.

In this case

$$a_1 = -a_0 = \sqrt{E} \tag{C.91}$$

μ_0 , the value of the threshold corresponding to $P_f = P_m$ is:

$$\mu_0 = \frac{a_1 + a_0}{2\sqrt{\sigma_n^2 T}} \tag{C.92}$$

and, based on (C.86), is obtained when $K=1$; it follows that:

$$P_E = Q\left(\frac{a_1 + a_0}{2\sqrt{\sigma_n^2 T}}\right) \quad (\text{C.93})$$

In the particular case (C.90)

$$\begin{cases} E_0 = E_1 = E \\ \rho = -1 \\ a_1 = -a_0 = \sqrt{E} \\ K = 1 \end{cases}$$

the bit error rate is:

$$P_E = Q\left(\sqrt{\frac{E}{\sigma_n^2 T}}\right) = Q\left(\sqrt{2\frac{E_b}{N_0}}\right) \quad (\text{C.94})$$

and show that the same BER is obtained with 3dB less E_b/N_0 than in the case of one signal (0, $s(t)$) - see relation (C.62).

References

- [1] Kay, S.M.: Fundamentals of Statistical Signal Processing, Detection Theory. Prentice-Hall, Englewood Cliffs (1998)
- [2] Papoulis, A.: Probability, Random Variables and Stochastic Processes, 3rd edn. McGraw-Hill Companies, New York (1991)
- [3] Sklar, B.: Digital Communications. Prentice-Hall, Englewood Cliffs (1988)
- [4] Spataru, A.: Fundaments de la théorie de la transmission de l'information. Presse Polytechnique Romandes, Lausanne (1987)
- [5] Stanley, W.D.: Electronic Communications Systems. Prentice-Hall, Englewood Cliffs (1982)
- [6] Van Trees, H.L.: Detection, Estimation and Modulation Theory, Part 1. J. Wiley, New York (1968)
- [7] Xiong, F.: Digital Modulation Techniques. Arch House, Boston (2000)

Appendix D: Synthesis Example

We think that an example, trying to synthesize the main processings exposed in the present book: source modelling, compression and error protection, is helpful for those who already acquired the basics of information theory and coding and also for the very beginners, in order to understand the logic of processing in any communication/ storage system. Such examples are also suitable for examination from the topic above.

The message VENI_VIDI_VIVI is compressed using a binary optimal lossless algorithm.

1. Calculate the efficiency parameters of the compression and find the binary stream at the output of the compression block.
2. Which are the quantity of information corresponding to letter V, the quantity of information per letter and the information of the whole message?
3. Which is the quantity of information corresponding to a zero, respectively a one of the encoded message? Show the fulfilment of lossless compression relation.

The binary stream from 1, assumed to have 64kbps, is transmitted through a BSC with $p = 10^{-2}$.

4. Determine the channel efficiency and the required bandwidth in transmission.

Assume that before transmission/storage, the binary stream from 1 is error protected using different codes. Find the first non-zero codeword and the required storage capacity of the encoded stream. The error-protection codes are:

5. Hamming group with $m = 4$ (information block length) of all three varieties (perfect, extended and shortened).
6. Cyclic, one error correcting code, with $m=4$, using LFSR.
7. BCH with $n = 15$ and $t = 2$ (number of correctable errors).
8. RS with $n = 7$ and $t = 2$.
9. Convolutional non-systematic code with $R=1/2$ and $K=3$.

All the answers need to be argued with hypothesis of the theoretical development and comments are suitable.

Solution

A block scheme of the presented processing is given in Fig. D.1.

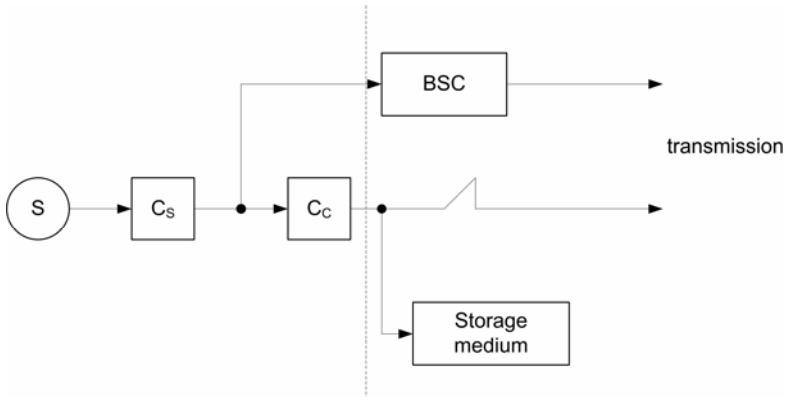


Fig. D.1 Block-scheme of processing where:

- S represents the message (its statistical model)
- C_S - compression block
- C_C - channel coding block (error control coding)

1. The message VENI_VIDI_VICI, under the assumption of being *memoryless* (not true for a language) is modelled by the PMF:

$$S: \begin{pmatrix} V & E & N & I & \bar{} & D & C \\ \frac{3}{14} & \frac{1}{14} & \frac{1}{14} & \frac{5}{14} & \frac{2}{14} & \frac{1}{14} & \frac{1}{14} \end{pmatrix}$$

For compression is chosen the binary Huffman static algorithm which fulfils the requirements: lossless, optimal, binary and PMF known (previously determined).

As described in 3.7.2 (Remarks concerning Huffman algorithm) the obtained codes are not unique, meaning that distinct codes could be obtained, but all ensure the same efficiency ($\bar{1}$).

In what follows two codes are presented, obtained using the same S and algorithm.

(a)							
s_i	p_i	c_i	R_1	R_2	R_3	R_4	R_5
I	5/14	1	5/14	5/14	5/14	5/14	9/14 0
V	3/14	000	3/14	3/14	4/14	5/14 } ₀₀	5/14 ①
-	2/14	001	2/14	2/14	3/14 } ₀₀₀	4/14 } ₀₁	
E	1/14	0110	2/14	2/14 } ₀₁₀	2/14 } ₀₀₁		
N	1/14	0111	1/14 } ₀₁₁₀	2/14 } ₀₁₁			
D	1/14	0100	1/14 } ₀₁₁₁				
C	1/14	0111					

(b)							
s_i	p_i	c_i	R_1	R_2	R_3	R_4	R_5
I	5/14	00	5/14	5/14	5/14	5/14	9/14 0
V	3/14	10	3/14	3/14	4/14	5/14 } ₀₀	5/14 1
-	2/14	11	2/14	2/14	3/14 } ₁₀	4/14 } ₀₁	
E	1/14	0110	2/14	2/14 } ₀₁₀	2/14 } ₁₁		
N	1/14	0111	1/14 } ₀₁₁₀	2/14 } ₀₁₁			
D	1/14	0100	1/14 } ₀₁₁₁				
C	1/14	0101					

$$\bar{l}_a = \sum_{i=1}^7 p_i l_i = \frac{5}{14} + \frac{3}{14} \cdot 3 + \frac{2}{14} \cdot 3 + 4 \cdot \frac{1}{14} \cdot 4 = \frac{5+9+6+16}{14} = \frac{36}{14} = 2.57$$

$$\bar{l}_b = \frac{5}{14} \cdot 2 + \frac{3}{14} \cdot 2 + \frac{2}{14} \cdot 2 + \frac{1}{14} \cdot 4 = \frac{36}{14} = 2.57$$

The output of the compression block (C_S), using (a) code is:

000 0110 0111 1 001 000 1 0100 1 001 000 1 0111 1
 V E N I _ V I D I _ V I C I

Efficiency parameters: coding efficiency η and compression ratio R_C , given by (3.46), respectively (3.49) are:

$$\eta = \frac{\bar{l}_{\min}}{\bar{l}} = \frac{H(S)}{2.57} = \frac{2.49}{2.57} \cong 0.968 \quad (97.7\%)$$

$H(S)$, source entropy, is given by (2.12):

$$H(S) = -\sum_{i=1}^7 p_i \log_2 p_i = 2.49 < H_{\max}(S) = D(S) = \log_2 7 = 2.80$$

Remark: Always is good to check the calculus and to compare to limits that are known and easy to compute.

$$R_C = \frac{l_u}{l}, \text{ where } l_u \text{ is the length in uniform encoding and is obtained using (3.58.a)}$$

$$l_u = \frac{\log_2 M}{\log_2 m} = \frac{\log_2 7}{\log_2 2} = \frac{2.80}{1} \approx 3 \text{ (the first superior integer)}$$

$$R_C = \frac{3}{2.57} \approx 1.2$$

2. Under the assumption of a memoryless source, we have:

- the self information of letter V, according to (2.11) is:

$$i(V) = -\log_2 p(V) = -\log_2 \frac{3}{14} \approx 2.2 \text{ bits}$$

- the average quantity of information per letter is $H(S)=2.49$ bits/letter
- the information of the whole message (14 letters) can be obtained in several ways; the simplest way is to multiply the number of letters ($N=14$) of the message with the average quantity of information per letter ($H(S)$), using the additivity property of the information.

$$I_M = N \times H(S) = 14 \text{ letters} * 2.49 \text{ bits/letters} = 34.86 \text{ bits}$$

Another possibility, longer as calculus, is to calculate the self information of each letter and then to multiply with the number of occurrence in the message. The result will be the same, or very close (small differences occur because of the logarithm calculus, on the rounding we do). The reader is invited to check it.

3. By compression the message (source S) is transformed in a secondary binary source (X). Under the assumption that this new source is also memoryless, we can model it statistically with the PMF:

$$X = \begin{pmatrix} 0 & 1 \\ p(0) & p(1) \end{pmatrix}, p(0) + p(1) = 1$$

$$p(0) = \frac{N_0}{N_0 + N_1}$$

$$p(1) = \frac{N_1}{N_0 + N_1}$$

where N_0 , respectively N_1 represent the number of “0”s, respectively “1”s in the encoded sequence. Counting on the stream determined at 1, we have:

$$\left. \begin{array}{l} p(0) = \frac{20}{36} \cong 0.55 \\ p(1) = \frac{16}{36} \cong 0.45 \end{array} \right\} \Rightarrow X : \begin{pmatrix} 0 & 1 \\ 0.55 & 0.45 \end{pmatrix}$$

- $p(0) \approx p(1) = 0.5$ the condition of statistical adaptation of the source to the channel obtained by encoding is only approximately obtained; the encoding algorithm, an optimal one, is introducing, by its rules, a slight memory.

- the information corresponding to a zero is:

$$i(0) = -\log_2 p(0) = -\log_2 0.55 = 1.86 \text{ bits}$$

- based on the same reason, as in 2, the information of the whole encoded message is:

$$I_{Me} = (N_0 + N_1)H(X)$$

$$H(X) = -p(0)\log_2 p(0) - p(1)\log_2 p(1) = -0.55\log_2 0.55 - 0.45\log_2 0.45 = \\ = 0.4744 + 0.5184 = 0.9928 \text{ bits/binary simbol}$$

$$I_{Me} = 36 * 0.99 = 35.64 \text{ bits}$$

Remarks

The equality (approximation in calculus give basically the difference between them) between $35.14 = I_M \approx I_{Me} = 35.64$ shows the conservation of the entropy in loss-less compression. The same condition can be expressed using relation (3.47):

$$H(S) = \bar{H}(X)$$

$$2.49 = 2.57 \cdot 0.99 = 2.54$$

4. Channel efficiency is expressed using (2.66):

$$\eta_C = \frac{I(X;Y)}{C}$$

where the transinformation $I(X;Y)$, can be obtained using (2.58):

$$I(X;Y) = H(Y) - H(Y/X)$$

Taking into account that the average error $H(Y/X)$ for a BSC was calculated in (2.70):

$$H(Y/X) = -p\log_2 p - (1-p)\log_2 (1-p) = \\ = -(0.01\log_2 0.01 + 0.99\log_2 0.99) = 0.0822 \text{ bits/symbol}$$

$H(Y)$ requires the knowledge of PMF. It can be obtained in more ways (see 2.8.1). The simplest in this case is using (2.31):

$$P(Y) = P(X)P(Y/X) = \begin{bmatrix} p(0) & p(1) \end{bmatrix} \begin{bmatrix} 1-p & p \\ p & 1-p \end{bmatrix} = \\ = \begin{bmatrix} 0.55 & 0.45 \end{bmatrix} \begin{bmatrix} 0.99 & 0.01 \\ 0.01 & 0.99 \end{bmatrix} = \begin{bmatrix} 0.549 & 0.451 \end{bmatrix}$$

$$H(Y) = -0.549\log_2 0.549 - 0.451\log_2 0.451 = 0.4750 + 0.5181 = 0.9931$$

It follows that

$$I(X;Y) = H(Y) - H(Y/X) = 0.9931 - 0.0822 = 0.9109$$

Capacity of BSC is given by (2.71):

$$\begin{aligned} C_{\text{BSC}} &= 1 + p \log_2 p + (1-p) \log_2 (1-p) \\ &= 1 - H(Y/X) = 0.9178 \text{ bits/symbol} \end{aligned}$$

Now the channel efficiency can be obtained:

$$\eta_C = \frac{I(X;Y)}{C} = \frac{0.9109}{0.9178} \approx 0.993$$

The required bandwidth in transmission assuming base-band transmission, depends of the type of the code is used. In principle there are two main types, concerning BB coding (NRZ- and RZ- see 5.12)

Using the relation (2.28), real channels, we have:

$$\begin{aligned} B \cong 0.8M &= \begin{cases} 0.8 * 64 * 10^3 & \text{(for NRZ)} \\ 2 * 0.8 * 64 * 10^3 & \text{(for RZ)} \end{cases} \\ &= \begin{cases} 51.2 \text{Khz} & \text{(for NRZ - BB coding)} \\ 102.4 \text{Khz} & \text{(for RZ - BB coding)} \end{cases} \end{aligned}$$

For channel encoding, the binary information from 1 is processed according to the type of the code.

$$\begin{array}{c} 0 \ 000110011110010001010011001000101111 \\ \text{MSB} \end{array}$$

5. Encoding being with *Hamming group code* with $m=4$, the whole input stream ($N=36$ bits) is split in blocks of length $m=4$, with the MSB first in the left. If necessary, padding (supplementary bits with 0 values) is used.

In our case: $N_H = \frac{N}{m} = \frac{36}{4} = 9$ codewords (no need for padding).

• *Perfect Hamming* with $m=4$ is given by (5.71), (5.72), (5.73) and (5.74).

$$n = 2^k - 1 = m + k = 2^k - 1 \text{ where } n = m + k.$$

For $m=4$ it follows that $k=3$ and $n=7$. The codeword structure is:

$$\begin{aligned} v &= [c_1 \ c_2 \ a_3 \ c_4 \ a_5 \ a_6 \ a_7] \\ H &= \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \end{aligned}$$

The encoding relations, according to (5.38) are:

$$\begin{aligned} H v^T = 0 &\Rightarrow c_4 = a_5 \oplus a_6 \oplus a_7 \\ &c_2 = a_3 \oplus a_6 \oplus a_7 \\ &c_1 = a_3 \oplus a_5 \oplus a_7 \end{aligned}$$

The first codeword, corresponding to the first 4 bits block, starting from the right to left: $i = [1111] = [a_7 a_6 a_5 a_3]$

$$v = [1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1]$$

The required capacity to store the encoded stream is:

$$C_H = N_H \times n = 9 \times 7 = 63 \text{ bits}$$

- *Extended Hamming* for $m=4$ is:

$$v^* = [c_0 \ c_1 \ c_2 \ a_3 \ c_4 \ a_5 \ a_6 \ a_7]$$

$$H^* = \begin{bmatrix} 0 & :0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & :0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & :1 & 0 & 1 & 0 & 1 & 0 & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$H^* \cdot v^{*T} = 0$ is giving c_4, c_2, c_1 as for the perfect code and

$$c_0 = c_1 \oplus c_2 \oplus a_3 \oplus c_4 \oplus a_5 \oplus a_6 \oplus a_7 = 1,$$

and thus, the first non zero extended codeword is:

$$v^* = [1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1]$$

The required capacity to store the encoded stream is:

$$C_H^* = N_H \times n^* = 9 \times 8 = 72 \text{ bits}$$

- *Shortened Hamming*, with $m=4$ is (see Example 5.8), obtained starting from the perfect Hamming with $n=15$ and deleting the columns with even “1”s:

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

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

The codeword structure is:

$$v_S = [c_1 \ c_2 \ c_3 \ a_4 \ c_5 \ a_6 \ a_7 \ a_8]$$

and the encoding relations, obtained from

$$H_S v_S = 0$$

are:

$$c_5 = a_6 \oplus a_7 \oplus a_8$$

$$c_3 = a_4 \oplus a_7 \oplus a_8$$

$$c_2 = a_4 \oplus a_6 \oplus a_8$$

$$c_1 = a_4 \oplus a_6 \oplus a_7$$

For the four information bits: $i = [a_4 \ a_6 \ a_7 \ a_8] = [1 \ 1 \ 1 \ 1]$, the first codeword is:

$$v_S = [1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1]$$

The storage capacity of the encoded stream is:

$$C_{H_S} = N_H \times n_S = 9 \times 8 = 72 \text{ bits}$$

6. For a *cyclic one error correcting code*, meaning a BCH code of $m=4, t=1$, from Table. 5.8 we choose the generator polynomial:

$$g = [1 \ 3] = [001011] \rightarrow g(x) = x^3 + x + 1$$

The block scheme of the encoder, using LFSR is (see Fig. 5.13) given in Fig. D.2.

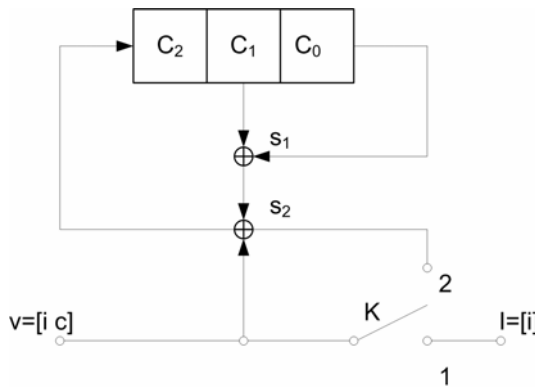


Fig. D.2 Block scheme of cyclic encoder with LFSR with external modulo two sumators and $g(x) = x^3 + x + 1$

The structure of the codeword is:

$$v = [i \ c] = \underbrace{[a_6 \ a_5 \ a_4 \ a_3]}_{i:m=4} \underbrace{[a_2 \ a_1 \ a_0]}_{c:k'=3}$$

For the first four information bits: $i=[1\ 1\ 1\ 1]$ the operation of the LFSR is given in the following table:

t_n			t_{n+1}			t_n
C_k	K	i	C_2	C_1	C_0	v
1	1	1	1	0	0	1
2		1	1	1	0	1
3		1	0	1	1	1
4		1	1	0	1	1
5	2		0	1	0	1
6			0	0	1	1
7			0	0	0	1

Concerning the number of codewords for the binary input from 1, this is the same as for Hamming codes, m being the same.

The storage capacity is:

$$C_{BCH1} = N_{BCH1} \times n_{BCH1} = 9 \times 7 = 63 \text{ bits}$$

7. For BCH , $n=15, t=2$ we choose from Table. 5.8: $g = [7\ 2\ 1] = [111010001]$

$$g(x) = x^8 + x^7 + x^6 + x^4 + 1 \Rightarrow k' = 8 \Rightarrow m = 7$$

A systematic structure is obtained using the algorithm described by relation (5.98); we choose, as information block, the first $m=7$ bits, starting from right to left:

$$i = [\underset{\text{MSB}}{0\ 1\ 0\ 1\ 1\ 1\ 1}]$$

- $i(x) = x^5 + x^3 + x^2 + x + 1$
- $x^{k'}i(x) = x^8(x^5 + x^3 + x^2 + x + 1) = x^{13} + x^{11} + x^{10} + x^9 + x^8$
- $\frac{x^{k'}i(x)}{g(x)} = q(x) + \text{rem} \frac{x^{k'}i(x)}{g(x)} = \underbrace{x^5 + x^4 + x^3 + x^2 + 1}_{q(x)} + \underbrace{x^5 + x^3 + x^2 + 1}_{\text{rem} \frac{x^{k'}i(x)}{g(x)}}$
- $v(x) = x^{k'}i(x) + \text{rem} \frac{x^{k'}i(x)}{g(x)} = q(x) \cdot g(x) = x^{13} + x^{11} + x^{10} + x^9 + x^8 + x^5 + x^3 + x^2 + 1$

In matrix expression: $v = [\underset{\text{MSB}}{0\ 1\ 0\ 1\ 1\ 1\ 1} : 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1]$.

The same result can be obtained using a LFSR with the characteristic polynomial $g(x)$. The reader is invited to check this way too.

The number of codewords corresponding to the stream 1 for $m=7$ is:

$$N_{\text{BCH2}} = \frac{N}{m} = \frac{36}{7} = 5.1 \text{ meaning that padding is necessary to obtain 6}$$

codewords: the required padding bits of “0” are 6.

The storage capacity is:

$$C_{\text{BCH2}} = N_{\text{BCH2}} \times n = 6 \times 15 = 90 \text{ bits}$$

8. RS with $n=7$ and $t=2$

The dimensioning of this code was given in Example 5.19.

$n = 2^k - 1 = 7 \Rightarrow k = 3 \Rightarrow \text{GF}(2^3)$ is the corresponding Galois field of this code and it is given in Appendix A.10.

For $t = 2$ (the number of correctable symbols), the generator polynomial is:

$$g(x) = (x + \alpha)(x + \alpha^2)(x + \alpha^3)(x + \alpha^4) = x^4 + \alpha x^3 + x^2 + \alpha x + \alpha^3$$

which means that $k' = 4$ and it represents the corresponding number of control characters. It follows that the number of information characters is:

$$m = n - k' = 7 - 4 = 3$$

We notice that each character is expressed in k bits, k being the extension of the Galois field $\text{GF}(2^k)$, which is in our case 3. It means that for encoding we need blocks of:

$$m \times k = 3 \times 3 = 9 \text{ bits}$$

The first 9 bits corresponding to the binary stream from 1 are:

$$\mathbf{i} = [0 \ 0 \ 0 : 1 \ 0 \ 1 : 1 \ 1 \ 1]$$

\uparrow
 MSC

Using the table giving $\text{GF}(2^3)$ from A.10 we identify:

$$000 \rightarrow \mathbf{0}$$

$$101 \rightarrow \alpha^6$$

$$111 \rightarrow \alpha^5$$

Now we apply the algebraic encoding for systematic codes (5.98), with the observation that the calculus is in $\text{GF}(2^3)$.

- $\mathbf{i}(x) = \alpha^6 x + \alpha^5$
- $x^{k'} \mathbf{i}(x) = x^4 (\alpha^6 x + \alpha^5) = \alpha^6 x^5 + \alpha^5 x^4$
- $\frac{x^{k'} \mathbf{i}(x)}{\mathbf{g}(x)} = \underbrace{\alpha^6 x + \alpha^4}_{\mathbf{q}(x)} + \underbrace{\alpha x^3 + \alpha^5 x^2 + \alpha^3 x + 1}_{\text{rem} \frac{x^{k'} \mathbf{i}(x)}{\mathbf{g}(x)}}$
- $\mathbf{v}(x) = x^{k'} \mathbf{i}(x) + \text{rem} \frac{x^{k'} \mathbf{i}(x)}{\mathbf{g}(x)} = \alpha^6 x^5 + \alpha^5 x^4 + \alpha x^3 + \alpha^5 x^2 + \alpha^3 x + 1$

In a matrix representation, \mathbf{v} is:

$$\begin{aligned} \mathbf{v} &= [\mathbf{0} \quad \alpha^6 \quad \alpha^5 \quad \alpha \quad \alpha^5 \quad \alpha^3 \quad \mathbf{1}] - \text{in } \text{GF}(2^3) \\ &= [0 \quad 7 \quad 6 \quad 2 \quad 6 \quad 4 \quad 1] - \text{in decimal} \\ &= \left[\begin{array}{cccccccc} 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{array} \right]_{\text{LSB}} - \text{in binary} \end{aligned}$$

The number of codewords, corresponding to the binary stream 1 is:

$$N_{\text{RS}} = \frac{N}{m \times k} = \frac{36}{3 \times 3} = 4 \quad (\text{no necessary for padding})$$

The required capacity for storage of the encoded stream 1 is:

$$C_{\text{RS}} = N_{\text{RS}} \times n_{\text{RS}} \times k = 4 \times 7 \times 3 = 84 \text{ bits}$$

9. Non-systematic convolutional code with $R=1/2$ and $K=3$

The dimensioning and encoding of the code is presented in 5.92.

- $R = \frac{1}{2} \Rightarrow n = 2$: two generator polynomials are required, from which at least one need to be of degree $K-1=2$

We choose:

$$g^{(1)}(x) = 1 + x + x^2$$

$$g^{(2)}(x) = 1 + x$$

- the information stream is the binary stream from 1 ending with four “0”s, which means that with trellis termination.
- the simplest way to obtain the encoded stream is to use the SR implementation of the encoder (Fig. D.3)

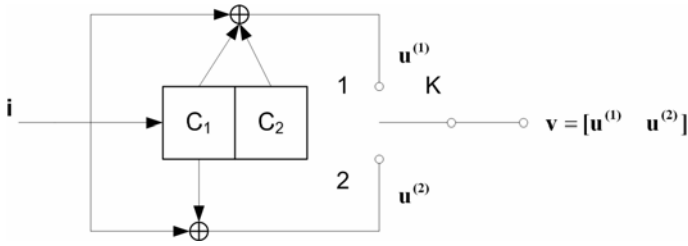


Fig. D.3 Non-systematic convolutional encoder for $R=1/2$ and $K=3$ ($g^{(1)}(x) = 1 + x + x^2$, $g^{(2)}(x) = 1 + x$)

The result obtained by encoding with the non-systematic convolutional encoder from Fig. D.3, for the information stream:

$i = \left[\begin{array}{l} 0 \\ \text{MSB} \end{array} \quad 0001100111110010001010011001000101111 \right]$ is shown in the table D1.

The required capacity to store the encoded stream from 1 is:

$$C_{\text{conv}} = N \times n = 36 \times 2 = 72 \text{ bits}$$

Remark

In our example only the encoding was presented, but we assume that the reverse process, decoding, is easily understood by the reader. Many useful examples for each type of processing found in chapter 3 and 5 could guide the full understanding.

Table D.1 Operation of the encoder from Fig. D.1 for the binary stream i (the output of the compression block)

t_n		t_{n+1}		t_n	
C_k	i	C_1	C_2	V	
				$u^{(1)}$	$u^{(2)}$
1	1	1	0	1	1
2	1	1	1	0	0
3	1	1	1	1	0
4	1	1	1	1	0
5	0	0	1	0	1
6	1	1	0	0	1
7	0	0	1	1	1
8	0	0	0	1	0
9	0	0	0	0	0
10	1	1	0	1	1
11	0	0	1	1	1
12	0	0	0	1	0
13	1	1	0	1	1
14	0	0	1	1	1
15	0	0	0	1	0
16	1	1	0	1	1
17	0	0	1	1	1
18	1	1	0	0	1
19	0	0	1	1	1
20	0	0	0	1	0
21	0	0	0	0	0
22	1	1	0	1	1
23	0	0	1	1	1
24	0	0	0	1	0
25	1	1	0	1	1
26	1	1	1	0	0
27	1	1	1	1	0
28	1	1	1	1	0
29	0	0	1	0	1
30	0	0	0	1	0
31	1	1	0	1	1
32	1	1	1	0	0
33	0	1	1	0	1
34	0	0	1	0	1
35	0	0	0	1	0
36	0	0	0	0	0

Subject Index

- absolute optimal codes 81
- absolute redundancy 13, 218
- absorbent 42, 43
- absorbent chain 42
- acceptable loss 75
- access control 124
- active monitoring 182
- adaptive chosen plaintext attack 126
- addition circuit over GF(2k) 309
- ad-hoc compression 95
- AES (Advanced Encryption Standard) 147
- Alberti cipher 122, 132
- algebra 122, 256, 389,
- algebraic decoding for RS codes 298
- algorithm 85-96, 100-106, 123-126, 147-150
- almost perfect codes 228-229
- ambiguity attacks 178
- AMI (Alternate Mark Inversion) 378
- amino-acids 77, 78, 79
- amplitude resolution 31, 32, 37
- anonymity 124
- arabian contribution 121
- arbitrated protocol 179, 183
- arithmetic coding algorithm 107
- ARQ - automatic repeat request 213-215, 228, 267
- asymmetric (public-key) 125
- attack 123, 125-127, 129, 136, 151, 169, 177-181, 194
- audiosteganography 167
- authentication 122, 124, 153, 158-160, 164, 183-185
- authentication role 151
- authentication/ identification 124
- authorization 124
- automated monitoring 182
- average length of the codewords 80, 81, 94, 99
- backward recursion 357, 363, 368-371, 373
- base band codes / line codes 53, 383
- basic idea 85, 91-93, 95, 102, 107
- basic idea of differential PCM 109
- basic principle of digital communications 65
- BCH codes (Bose-Chauduri-Hocquenghem) 261, 269, 271, 296, 299, 420
- BCH encoding 262
- Berlekamp 256, 299
- Berlekamp - Massey algorithm 302
- bidirectional SOVA steps 358
- Binary N-Zero Substitution Codes 379
- Biphase codes 377, 378, 383
- Bipolar signalling 381, 382
- bit error rate (BER) 2, 36, 68, 72, 176, 229, 230, 233, 236-238, 342, 348, 374, 375, 381, 382, 438, 447, 453, 459
- bit or symbol synchronization 375
- bit synchronization capacity 377-379, 383
- B-L (Biphase Level-Manchester) 377
- block ciphers 137, 139, 147
- block codes 216, 219, 224, 228, 232, 253, 312-315, 324, 325, 341, 357-358
- block interleaving 340-342
- B-M (Biphase Mark) 378
- branch metric 351-353, 359, 364
- broadcast monitoring 166, 181, 184
- brute force attack 127, 129
- brute force attack dimension 136
- B-S (Biphase Space) 378
- burst error 49, 50, 216, 292, 340, 341
- burst error length 50, 340
- burst-error channels 50
- Caesar cipher 121, 128-130
- cancellation 124

- canonical forms 225, 227
- capacity 23-25, 30, 32-38, 53, 55, 107, 167, 218, 311, 461
- capacity boundary 35
- capacity formula 32
- CAST 149
- catastrophic 317
- catastrophic code 317
- central dogma of molecular biology 76, 191
- certification 124, 180
- channel 1-4
- channel absolute redundancy 24
- channel capacity 23-26, 29, 30, 32, 36
- channel efficiency 24, 27, 461
- channel redundancy 24
- channel relative redundancy 24
- Chapman-Kolmogorov relations 41
- characteristic matrix 154, 275-277, 280-282, 293
- chiffre indéchiffrable 134
- chosen ciphertext attack 126
- chosen plaintext attack 126
- chromosome 191, 194, 199-202
- cipher 121-123, 127-137, 147-151, 160-162, 198
- cipher disk 122, 128, 132
- ciphertext 150, 151, 123, 125-137, 197, 198, 200, 203
- ciphertext only attack 126
- classic cryptography 127
- CMI Coded Mark Inversion 377, 380, 383
- code distance 219, 222, 233, 241-246, 291, 313, 324, 325, 342
- code efficiency 90, 235, 375, 383
- code extension 250
- code generating matrix 224
- code shortening 251
- codes 53-55, 81-83
- codes for burst error 216, 340,
- codes for independent error control 216
- codeword 54, 55, 59, 75, 80-83
- codewords set 246, 248, 256
- coding 3, 48, 53
- coding efficiency 80, 81, 84, 85, 101, 463
- coding gain G_c 237, 333, 348
- coding rate 212, 313, 314, 347, 351, 370
- coding theory point of view 55, 81
- coding tree 55, 86, 91
- codon 77-79
- collusion by addition 181
- collusion by subtraction 181
- comma code 55, 100
- comparative analysis of line codes 382
- complexity and costs 375
- complexity of an attack 127
- compression 4, 5, 49, 53-56, 58-60, 68, 69, 71, 72, 80, 81, 85, 86, 88-93, 95-98, 99-101, 106, 107, 109-111, 116, 117, 129, 137, 172, 177, 213
- compression codes 53
- compression ratio 80, 81, 86, 90, 101, 111
- conditional entropy (input conditioned by output) / equivocation 19
- conditional entropy (output conditioned by input) / average error 19
- confidentiality and authentication 160
- confidentiality/ secrecy/ privacy 124
- confirmation 124, 214
- confusion 135, 137, 138, 178
- constant weight code (m, n) 255
- constraint (M) 313
- constraint length (K) 313, 317, 323, 326, 332, 334, 340, 341
- continuous (convolutional) codes 216
- continuous codes 314, 335
- control matrix 226, 230, 231, 239-242, 245, 259, 264, 265, 278, 282, 320
- control symbols 211, 216, 224, 226, 239, 258, 263, 264, 278, 291-293, 314-316, 335-337
- convolutional codes 312-314, 317, 320, 324, 325, 332-335, 340, 341, 346, 349
- convolutional interleaving 340-342
- copy protection 166, 168, 184
- copyright 165, 169-172, 178, 182, 183, 189
- copyright notice 182
- copyright protection 165-167, 169, 189
- correlator 172, 175, 176, 185, 450, 451, 457

- coset leader 232-234
- CRC (Cyclic Redundancy Check) 107, 267
- crest factor 70
- Cross parity check code 254
- cryptanalysis 122, 123, 126, 127, 129-131, 134-139, 156
- cryptanalyst 123, 126, 151, 163
- cryptanalytic attack 125
- cryptographer 123, 132, 133, 140
- cryptographic algorithm/ cipher 123
- cryptographic keys 123, 170, 190
- cryptography 121-123, 125, 127, 131-133, 135-137, 139, 158, 159, 167-169, 184, 190, 197, 200
- cryptologist 123, 134
- cryptology 122, 123
- cryptosystem 123, 124, 126, 135, 136, 148, 160, 161
- cumulated distance 327, 331
- cyclic codes for error detection 267

- d free (d_∞) 325
- data authentication 124, 164, 166
- data complexity 127
- decimal format of ASCII code 61
- decimal numeral system 57, 122
- decision noise 72-75, 446
- decoder with LFSR and external modulo 2 adders 287
- decoder with LFSR and internal modulo two adders 288
- decoding window 327, 331
- decryption/deciphering 123-127, 129, 134, 139-141, 145, 147-152, 162-164, 189, 198-203
- degeneracy 79
- Delay Modulation 378
- Delta modulation 111, 115, 117
- DES-X 148
- detection disabling attacks 178
- detection with and without original signal 169
- detection/correction capacity 218, 235
- dicode (twinned binary) 377, 379, 383
- differential BL 378
- differential coding 109, 376, 379
- diffusion 135, 137-139
- diffusion effect 135
- Digimarc 183, 184
- Digital Millennium Copyright Act 165, 184
- digital signature 124, 161, 164, 165, 184
- digital watermarking 164, 167, 189
- digram 131, 132
- direct orthogonal codes 336, 338, 339
- discrepancy 304
- discrete 1, 7, 8, 14-17, 27, 30, 44-47, 48, 49, 66, 93, 435
- distortion 5, 66, 67, 75, 178, 343, 346
- distribution of movie dailies 183
- DIVX Corporation 183
- DMI Differential Mode Inversion Code 377, 380, 383
- DNA (DeoxyriboNucleicAcid) 76, 77, 190-200
- DNA synthesiser 190
- DNA tiles 195-196
- DNA XOR with tiles 197
- DPCM codec 110

- electronic Codebook 150
- Encoder with LFSR 277
- Encoder with LFSR and internal modulo two adders 279, 281
- Encoders for systematic cyclic codes implemented with LFSR 277
- encoding equations 226
- encryption 53, 68, 69, 121, 123-125, 127, 152, 162-165, 184, 197-202
- encryption/ enciphering 123
- enrichment for the host signal 166
- entropic or lossless compression relation 80
- entropy of m-step memory source 44
- erasure channels 223
- erasure correction 223
- erasure symbol 223
- error control codes 53, 210, 216, 375, 383
- error correcting codes 172, 177, 209, 216, 218, 235, 238
- error correction 228, 240, 285-289
- Error correction and detection (ARQ hybrid systems) 214
- Error Correction Cyclic Decoder (Meggitt Decoder) 285, 286

- error density 50
- error detecting codes 216, 237, 254
- error detection 213, 222, 230, 253, 265-267, 282-284
- error detection capacity 375, 379, 380, 383
- error detection cyclic decoders 282
- error exponent 210
- error extensión 151
- error locator 270, 302
- error performance 375, 382, 383
- error performance of line codes 381
- error polynomial 270, 272, 299, 335
- error propagation 151
- error value 298, 303
- escape codeword 59
- Euler totient 162, 163
- Euler-Fermat Theorem 162
- even parity criterion 253, 254
- exons 76, 77, 191
- extended Hamming Codes 241
- extrinsic information 349

- fault-tolerant for point mutations 79
- FEAL 149
- feedback shift registers 153
- fingerprint 183
- fingerprinting 166, 183
- finite and homogenous Markov chain 39
- fixed – fixed 101
- fixed – variable 101
- Forward error correction (FEC) 213, 216, 228
- forward recursion 358, 359, 362, 368-370
- four fold degenerate site 79
- fragile watermark 166, 168
- frequency encoding 297

- gel electrophoresis 193, 198
- gene 76, 77, 191-194
- gene expression 76, 191-193
- generalized 129
- generates all the non-zero states in one cycle 276
- generator polynomial 154, 256, 258, 259, 261, 262, 266-268, 275, 287, 291, 293, 310, 314, 420, 421

- genetic code 76, 77-79, 119, 191
- genome 79, 192, 194
- geometric distortions 178
- geometric representation 217
- Gilbert Vernam 126, 132, 134
- Go-back N blocks system (GBN) 214, 215
- GOST 149
- granular noise 112-114
- graph 17, 26, 40, 83, 123, 321, 323, 324
- graphein 121

- half-duplex-type communication 214
- Hamming boundary 228
- Hamming distance 219, 324, 327, 351
- Hamming weight w 219, 325
- hard decision channel 332
- HAS - human audio system or HVS - human visual system 170
- HDBn (High Density Bipolar n) 380
- hexadecimal numeral system 57
- homogeneity 40
- hybridization 191, 193, 197, 198

- IBM attack 169, 171, 178-180
- IDEA 137, 148
- Identifiers 182
- immunity to polarity inversion 375, 383
- implicit redundancy 216, 218, 255
- independent 9, 20
- independent channel 22, 23
- indirect orthogonal codes 336, 338, 339
- information 1-5, 8-16, 21-23, 30
- information block 224, 267, 313, 469
- information representation codes 53, 55
- information symbols 211, 216, 224, 228, 239, 245, 258, 264, 277-279, 311, 314, 335
- Information Transmission System (ITS) 2
- initialization vector (IV) 151, 152
- input (transmission) alphabet 16
- input entropy 19, 20
- insertion 165, 168-178, 186, 187
- instantaneous code 55, 82, 83, 87
- Intentional attacks 168, 177, 181
- Interleaving 340-344

- International Standard Book Numbering 165
- International Standard Recording Code 165
- Interpolation 345, 346
- introns 76, 191
- inversion 169, 178, 268
- irreversible compression 117

- joint input – output entropy 19
- joint input–output 18
- joint probability mass function 18

- Kerckhoffs law 126
- key generation 141, 148, 171
- key management 201
- known plaintext attack 126
- Kraft inequality 82, 83, 85
- Kryptos 123

- length of a word 80
- LFSR 153
- LFSR with external modulo two adders 274
- LFSR with internal modulo two adders 276
- Limited 30, 32, 68, 75
- line codes, transmission modes, baseband formats/wave formats 374
- linear block code 224, 229, 255
- linear transformations in finite fields 137
- locators 270, 298, 299
- log - likelihood function 357

- main advantage 68
- main disadvantage 68, 182, 340
- majority criterion 211
- MAP decoding algorithm 350, 372
- Markov property* 40
- Masquerade 170, 174
- masquerading signal 170
- matrix representation 216, 227, 263, 402, 418
- maximum likelihood decoder 221, 350, 351
- maximum likelihood path (ML) 351, 357, 358
- McMillan theorem 83

- mean squared error ϵ 2
- medicine 39, 79, 166
- microarray 190, 193, 198
- Miller code 378
- minimal polynomial 260-263, 291, 404, 405-408
- MLD decoder 221
- modified Hamming Codes 240
- modified Huffman code 97, 98
- Modulation 3, 4, 15, 36, 49, 64, 66, 69, 109, 115, 116, 213, 348, 368-370, 374-378
- Moore law 127
- m-order memory source 38
- multilevel signalling 380, 382
- multiple DES 148
- Multiplication circuit over $GF(2^k)$ 310
- mute interpolation 346
- Mutual information 21

- necessary condition 113, 180, 228, 233, 317
- noise 1-4, 17-20, 22-33, 37, 49-53, 67-75, 97-102
- noiseless channel 20, 22, 23, 80
- noisy channels coding theorem 209
- Non-redundant coding 211
- Nonrepudiation 124
- NON-Return to Zero 377
- Nonsense 79, 123
- non-separable codes 216, 218
- non-systematic codes 216, 316, 317, 326
- non-uniform code 54, 59
- NRZ-L (Non Return to Zero Level) 377, 378
- NRZ-M (Non Return to Zero Mark) 377, 378
- NRZ-S (Non Return to Zero Space) 377, 378
- N-th order code distance 324
- Nucleotide 76, 191
- numeral system 56, 57
- Nyquist theorem 32, 68

- octal numeral system 57
- Odd parity criterion 253, 254
- Oligonucleotides 190

- one error correcting ($t=1$) Hamming group codes (perfect) 209, 228, 239
 One time pad 126
 one time pad (OTP) principle 136
 one-way hash 169
 operation mode 150
 optimal algorithms 85
 output entropy 19
 output(receiving) alphabet 16
 ownership deadlock 169
- P box 137-139
 parity check matrix 226
 parity-check 141, 226
 passive monitoring 182
 path metric 351-354, 357-359, 362, 363
 perceptual transparency 168, 169-171, 174, 185
 perfect secrecy 126, 135
 period of the characteristic matrix 276
 permutation 128, 137-143, 256
 perturbation 1, 20
 Peterson algorithm with Chien search 269, 270, 273, 299, 301, 312
 plaintext/ cleartext 123
 Polar signalling 382
 polyalphabetic substitution cipher 132, 133
 Polybius 121, 128, 130, 131
 Polybius square 130, 131
 polygramic cipher 131
 polymates 132
 polymerase chain reaction 192
 polynomial representation 217, 263, 264, 297, 310, 315, 335, 347, 402
 positional systems 57
 prediction gain 111
 primer 192
 primitive 154, 158, 260-262, 276, 290-297, 348, 395-401, 408-409, 417
 principle of spread-spectrum 172
 probe 193
 processing complexity (work factor) 127
 product cipher 135, 137, 139
 proof of ownership 182
 proteins 77-79, 191
 protocol of digital signature 161
- pseudo ternary line codes 378
 public key cryptography (PKC) 125, 159, 164, 201
- quantization noise 67, 69, 75, 111, 113
 quantization noise limited region 75
- randomness and non-repeating OTPs 200
 rate distortion theory 75
 ratio 14, 26, 35, 50, 70, 84, 214, 218, 347, 375, 453
 receiver 3, 4, 16, 20, 31, 53, 65, 102, 122, 161, 165, 167, 169, 211-214, 269, 341, 379-382, 457
 reception probability matrix 17
 reciprocal 21
 recognition sequence 192
 recombinant DNA technology 192
 recursive systematic convolutional codes 346
 redundancy 13, 24, 27, 28, 47, 79, 98, 107, 116, 117, 137, 210, 218, 235, 253, 255, 296, 378
 redundant coding 211
 redundant symbols 211, 212, 216, 218, 267
 Reed-Solomon (RS) codes 255, 295-301, 308-312, 342-346, 421
 reflected binary code 63
 region 99
 regular 41, 45, 99, 391, 410
 regulatory elements 191
 relative frequency analysis 130-132
 relative redundancy 13, 24, 28, 218, 253, 296, 314
 remarking concept 189
 removal attacks 178
 renaissance 122
 Return to Zero(RZ) 377
 reversible compression 117
 ribonucleic acid 190
 Rijndael 147
 Rivest – Shamir - Adleman (RSA) cipher 162, 164, 169
 RNA (RiboNucleicAcid) 77, 191-193, 199
 Robustness 101, 116, 168-171, 177, 181, 184, 185, 188

- rotor machines 128
- rounds 137, 141, 147-149
- RS coding and decoding using linear feedback shift registers LFSR 308
- rubber-hose cryptanalysis/ purchase key attack 126
- S box 137, 138
- Sacred Books 121
- SAFER 149
- Scramblers 378
- Scytalae 121
- Secret 121-125, 127, 136, 148-150, 159, 160, 162, 169, 184, 201
- Secret message transmission 166
- Security 122, 125-127, 130, 135, 137, 140, 158, 162-164, 174, 188
- selective repeat system (SR) 214
- separable codes 216, 218
- Shannon 5, 9-11, 23, 30, 32-35, 84, 85, 95, 106, 117, 126, 127, 134-137, 209-218, 346, 348
- Shannon – Fano algorithm 85
- Shannon first theorem or noiseless channels coding theorem 84
- Shannon limit for an AWGN 35
- Shannon second theorem 209, 210, 218, 238
- Shortened Hamming Codes 242
- Sign – value system 57
- Signal 1, 2, 4
- signal detection 381, 435, 438, 442
- signal spectrum 375, 383
- signal/noise ratio (SNR) ξ 2, 5, 30, 32, 33, 36, 66, 68-70, 74, 75, 112, 114, 115, 235-238, 370, 375, 382, 446, 447
- signature 122, 124, 161, 165, 167, 182-184
- Simple attacks 178
- Simple parity check codes 253
- sliding correlator 173, 176
- Slope-overload noise 112
- soft (decision) decoding 333, 334
- soft decision 333, 359, 363
- spectral efficiency (bandwidth efficiency) 35, 36
- standard array 232, 233, 248, 249
- state diagram 320, 351, 352
- static algorithms 91, 117
- stationary 42, 44-46, 93, 94
- stationary state PMF 42
- statistic compression 117
- statistical adaptation of the source to the channel 23, 53, 55, 81, 86, 464
- steganographia 122, 133
- steganography 122, 123, 133, 167, 190, 197, 198
- steganos 121
- step size 67
- Stop and wait (SW) system 214
- Storage 1-4, 53, 56, 58, 64, 69, 77, 80, 95, 117, 127, 166, 172, 189, 194, 213, 343
- storage complexity 127
- storage medium 1-4, 188
- stream ciphers 137, 153, 157
- substitution 122, 127-134, 137, 145, 377, 379, 380
- substitution ciphers 128-130, 132
- substitution line codes 379
- substitution Permutation Network 137, 139
- sufficient condition 82, 83, 222, 228
- survivor 327, 331, 351, 353, 354, 356, 359, 363, 364
- symmetric (conventional) 135
- symmetric channel 24, 25
- synchronization 3, 4, 53, 173, 176, 178, 187, 377-379, 383
- syndrome 227, 232-235, 239-242, 265, 269, 270, 282, 283, 291, 305, 338
- syndrome decoding (using lookup table) 232, 233-235, 248
- system 2, 11, 29, 36, 39, 42-47, 56, 57, 100, 111, 123, 126, 172, 213
- system using special abbreviations for repetitions of symbols 57
- systematic codes 216, 314, 317, 471
- t error correcting block code 230
- t errors correction 231
- t errors detection 231
- tabula recta 133, 134

- target 193
- termination 79
- textual Copyright notices 182
- the basic demands 185
- the collusion attack 178, 180
- The digital audio encoding system for CD (CIRC) 343
- the encoding law 225-227, 255, 259
- The Millennium Watermark System 184
- three STOP codons 79
- three-fold degenerate site 79
- Threshold decoding 313, 326, 334, 336, 338
- ticket concept 189
- tightly packed/lossless 228
- time encoding 296
- time resolution 31, 32, 37
- total efficiency 214
- transcription 76, 77, 101
- transinformation 21, 23, 25, 29, 55
- transition (noise) matrix 17, 27, 29
- translation 76-79, 187, 191
- transmission channel 1, 10, 26, 27, 30, 31, 235, 311
- transmission probability matrix 16
- transmission system 2, 26, 214, 365, 379, 435
- transposed matrix 227
- transposition 128, 137, 138, 143, 147, 288
- transposition ciphers 128
- Tree Diagram 321
- Trellis Diagram 323, 324, 365
- trellis termination 320, 349, 358, 471
- trial key 134, 135
- Trithemius cipher 132, 133
- trustworthy digital camera 184
- Turbo codes 36, 209, 333
- two-fold degenerate site 79
- unaesthetic 182
- unary system 57
- undetectable 227, 229, 237, 255, 346
- uniform code 54
- unintentional attacks 168, 177
- Unipolar signalling 382
- uniquely decodable code 55, 100
- upper bound of the error probability after decoding 230
- variable – fixed 101
- variable – variable 101
- vector representation 217
- very noisy channel (independent) 20
- videosteganography 167
- Vigénère ciphertext keyword 135
- Viterbi algorithm 313, 326, 327, 328, 333, 334, 348, 349, 373
- Watermark 165, 167-181, 182
- watermark detection 185, 187
- watermark extraction (detection) 165, 169-172, 175-180, 185
- watermark granularity 168
- watermark information 168-177, 180
- watermark insertion 165, 168-176, 186
- watermark payload 168, 173
- watermark signal 170, 171, 175
- watermarking 164-174, 176, 177, 180-185, 189
- watermarking techniques 166, 170, 171, 172
- weights distributio 229
- whitening technique 148, 149

Acronyms

AC – Alternating Current	CC – Channel Encoding Block
ACK – Acknowledge	CCITT – International Telegraph and Telephone Consultative Committee
ADC – Analog to Digital Converter/Conversion	CD – Compact Disc
ADCCP – Advanced Data Communication Control Procedure	CDMA – Code Division Multiplex Access
ADPCM – Adaptive DPCM	CFB – Cipher Feedback
AES – Advanced Encryption Standard	CIRC – Cross Interleaved Reed–Solomon Code
AHS – Audio Human Sense	CMI – Coded Mark Inversion
AMI – Alternate Mark Inversion	CR – Carriage Return
ANSI – American National Standard Institute	CRC – Cyclic Redundancy Check
AOC – Absolute Optimal Code	C _s – Source Encoding Block
APP – Aposteriori Probability	CU – Control Unit
ARQ – Automatic Repeat Request	DAC – Digital to Analog Converter/Conversion
ASCII – American Standard Code for Information Interchange	DC – Direct Current
AWGN – Additive White Gaussian Noise	DC _c – Channel Decoding Block
BB – Baseband	DCS – Source Decoding Block
BCD – Binary Coded Decimal	DCT – Discrete Cosine Transform
BCH – Bose–Chauduri– Hocquenghem code	DEL – Delete
BEC – Binary Erasure Channel	DES – Data Encryption Standard
BER – Bit Error Rate	DFT – Discrete Fourier Transform
BISYNC – Binary Symmetrical Communication Protocol	DM – Delta Modulation
B–L – Biphase Level–Manchester	DMI – Differential Mode Inversion Code
B–M – Biphase Mark	DMS – Discreet Memoryless Source
BMC – Biomolecular Computation	DNA – DeoxyriboNucleic Acid
BN – Binary Natural	DOC – Direct Orthogonal Codes
BNZS – Binary N–Zero Substitution Codes	DPCM – Differential PCM
BPSK – Binary Phase Shift Keying	dsDNA – double stranded DNA
B–S – Biphase Space	DVD – Digital Versatile Disc
BSC – Binary Symetric Channel	EBCDIC – Extended Binary Coded Decimal Interchange Code
CBC – Cipher Block Chaining	ECB – Electronic Codebook
	EOT – End of Transmission
	ESC – Escape

- FCS – Frame Checking Sequence
 FEC – Forward error correction
 FFT – Fast Fourier Transform
 FGK – Faller, Gallager and Knuth algorithm
 FLC – Frame Level Control
 FPGA – Filed Programmable Gate Array
 GBN – Go-back N blocks system
 GF – Galois Fields
 GOST – Gosudarstvenii Standard
 GSM – Group Special Mobile
 HAS – Human Audio System
 HDBn – High Density Bipolar n
 HDLC – High Data Link Control
 HDTV – High Definition Television
 HVS – Human Visual System
 IBM – International Business Machines Corporation
 IC – Instantaneous Code
 IC – Integrate Circuit
 IDEA – International Data Encryption Algorithm
 IEEE – Institute of Electrical and Electronics Engineers
 IFFT – Inverse FFT
 IOC – Indirect Orthogonal Codes
 IP – Inverse Permutation
 ISBN – International Standard Book Numbering
 ISDN – Integrated Services Digital Network
 ISI – InterSymbol Interference
 ISO – International Organization for Standardization
 ISR – Information Shift Register
 ISRC – International Standard Recording Code
 ITC – Information Theory and Coding
 ITS – Information Transmission System
 ITU – International Telecommunication Union
 JPEG – Joint Photographic Experts Group
 KPD – Kinetic Protection Device
 LF – Line Feed
 LFSR – Linear Feedback Shift Register
 LOG-MAP – Logarithmic MAP
 LP – Low Pass
 LPC – Linear Prediction Coding
 LPF – Low Pass Filter
 LSB – Least Significant Bit
 LSI – Large-Scale Integration
 LSR – Linear Shift Register
 LZ – Lempel-Ziv Algorithm
 LZW – Lempel – Ziv – Welch algorithm
 MAP – Maximum Aposteriori Probability algorithm
 MAX-LOG-MAP – Maximum LOG-MAP
 MD-4/5 – Message-Digest algorithm 4/5
 ML – Maximum Likelihood
 MLD – Maximum Likelihood Decoding
 MNP5 – Microcom Network Protocol 5
 MPEG – Moving Picture Experts Group
 MR – Memory Register
 mRNA – messenger RNA
 MSB – Most Significant Bit
 NAK – Not Acknowledge
 NASA – National Aeronautics and Space Administration
 NBCD – Natural BCD
 NBS – National Bureau of Standards
 NCBI – National Center for Biotechnology Information
 NIST –National Institute of Standards and Technology
 NRZ – Non-return to zero
 NRZ-L – Non Return to Zero Level
 NRZ-M – Non Return to Zero Mark)
 NRZ-S – Non Return to Zero Space
 NSA – National Security Agency
 OFB – Output Feedback
 OTP – One Time Pad
 OWH – One-Way Hash
 PAM – Pulse Amplitude Modulation

PC – Personal Computer	SHA – Secure Hash Algorithm
PCI – Peripheral Component Interconnect	SNR – Signal to Noise Ratio
PCM – Pulse Code Modulation	SOVA – Soft Output Viterbi Algorithm
PCR – Polymerase Chain Reaction	SP – Space
PEM – Privacy Enhanced Mail	SPN – Substitution Permutation Network
PGP – Pretty Good Privacy	SPOMF – symmetrical phase only filtering –
PKC – Public Key Cryptography	SR – Syndrome Register
PMF – Probability Mass Function	ssDNA – single stranded DNA
PS – Signal Power	SW – Stop and Wait
PSD – Power Spectral Density	SWIFT – Society for Worldwide Interbank Financial Telecommunication
QPSK – Quaternary Phase Shift Keying	T-DES – Triple DES
RAM – Random Access Memory	TC – Turbo Codes
RC – Compression Ratio	TDMA – time division multiplex access
RDFT – Reverse DFT	TLD – Threshold Logic Device
Return to Zero (RZ)	tRNA – transfer RNA
RF – Radio Frequency	TV – Television
RLC – Run Length Coding	UDC – Uniquely Decodable Code
RNA (RiboNucleicAcid)	VA – Viterbi Algorithm
ROM – Read Only Memory	VBI – vertical blanking interval
RS – Reed–Solomon code	VHS – Video Human Sense
RSA – Rivest – Shamir – Adleman	VLSI – Very Large–Scale Integration
RSC – Recursive Systematic Convolutional codes	WM – watermark
S/MIME – Secure/ Multiple purpose Internet Mail Extension	XPD – protect portable devices
SAFER – Secure and Fast Encryption Routine	ZIP – ZIP file format
SDLC – Synchronous Data Link Control	

“I grow old learning something new everyday.”

Solon



Monica BORDA received the Ph.D degree from “Politehnica” University of Bucharest, Romania, in 1987. She has held faculty positions at the Technical University of Cluj-Napoca (TUC-N), Romania, where she is an Advisor for Ph.D. candidates since 2000. She is a Professor of Information Theory and Coding, Cryptography and Genomic Signal Processing with the Department of Communications, Faculty of Electronics, Telecommunications and Information Technology, TUC-N. She is also the Director of the Data Processing and Security Research Center, TUC-N. She has conducted research in coding theory, nonlinear signal and image processing, image watermarking, genomic signal processing and computer vision having authored and coauthored more than 100 research papers in referred national and international journals and conference proceedings. She is the author and coauthor of five books. Her research interests are in the areas of information theory and coding, signal, image and genomic signal processing.