

Forum for Interdisciplinary Mathematics

K. Deergha Rao

Channel Coding Techniques for Wireless Communications

Second Edition



Springer

Forum for Interdisciplinary Mathematics

Editor-in-Chief

P. V. Subrahmanyam, Department of Mathematics, Indian Institute of Technology Madras, Chennai, Tamil Nadu, India

Editorial Board

Yogendra Prasad Chaubey, Department of Mathematics and Statistics, Concordia University, Montreal, QC, Canada

Jorge Cuellar, Principal Researcher, Siemens AG, München, Bayern, Germany

Janusz Matkowski, Faculty of Mathematics, Computer Science and Econometrics, University of Zielona Góra, Zielona Góra, Poland

Thiruvenkatachari Parthasarathy, Chennai Mathematical Institute, Kelambakkam, Tamil Nadu, India

Mathieu Dutour Sikirić, Institute Rudjer Bošković, Zagreb, Croatia

Bhu Dev Sharma, Forum for Interdisciplinary Mathematics, Meerut, Uttar Pradesh, India

Forum for Interdisciplinary Mathematics is a Scopus-indexed book series. It publishes high-quality textbooks, monographs, contributed volumes and lecture notes in mathematics and interdisciplinary areas where mathematics plays a fundamental role, such as statistics, operations research, computer science, financial mathematics, industrial mathematics, and bio-mathematics. It reflects the increasing demand of researchers working at the interface between mathematics and other scientific disciplines.

More information about this series at <http://www.springer.com/series/13386>

K. Deergha Rao

Channel Coding Techniques for Wireless Communications

Second Edition



Springer

K. Deergha Rao
Department of ECE
Vasavi College of Engineering
(Autonomous college affiliated
to Osmania University)
Hyderabad, Telangana, India

ISSN 2364-6748 ISSN 2364-6756 (electronic)
Forum for Interdisciplinary Mathematics
ISBN 978-981-15-0560-7 ISBN 978-981-15-0561-4 (eBook)
<https://doi.org/10.1007/978-981-15-0561-4>

1st edition: © Springer India 2015

2nd edition: © Springer Nature Singapore Pte Ltd. 2019

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, 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.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721,
Singapore

मातृभ्यो नमः
पितृभ्यो नमः
गुरुभ्यो नमः

To

*My Parents Boddu and Dalamma,
My Beloved Wife Sarojini,
and My Mentor Prof. M.N.S. Swamy*

Preface

Lives of people have tremendously changed in view of the rapid growth of mobile and wireless communication. Channel coding is the heart of digital communication and data storage. Traditional block codes and conventional codes are commonly used in digital communications. To approach the theoretical limit for Shannon's channel capacity, the length of a linear block code or constant lengths of convolutional codes have to be increased, which in turn makes the decoder complexity higher and may render it physically unrealizable. The powerful turbo and LDPC codes approach the theoretical limit for Shannon's channel capacity with feasible complexity for decoding. MIMO communications is a multiple antenna technology which is an effective way for high-speed or high-reliability communications. The MIMO can be implemented by space-time coding. Recently, a new channel coding technique, namely polar codes, has emerged as one of the channel coding techniques for fifth-generation (5G) wireless communications, and it has been recommended by third-generation partnership project (3GPP) as a channel coding scheme for enhanced mobile broadband (eMBB) in 5G systems. However, the market lacks a book which can serve as a textbook for graduate and undergraduate students on channel coding techniques.

This book includes illustrative examples in each chapter for easy understanding of coding techniques. An attractive feature of this book is the inclusion of MATLAB-based examples with codes encouraging readers to implement them on their personal computers and become confident of the fundamentals by gaining more insight into coding theory. In addition to the problems that require analytical solutions, MATLAB exercises are introduced to the reader at the end of each chapter.

This book is divided into 13 chapters. Chapter 1 introduces the basic elements of a digital communication system, statistical models for wireless channels, capacity of a fading channel, Shannon's noisy channel coding theorem, and the basic idea of coding gain. Chapter 2 gives an overview of the performance analysis of different modulation techniques and also deals with the performance of different diversity combining techniques in a multichannel receiver. Chapter 3 introduces Galois fields and polynomials over Galois fields. Chapter 4 covers linear block codes including

RS codes because of their popularity in burst error correction in wireless networks. Chapter 5 discusses the design of a convolutional encoder and Viterbi decoding algorithm for the decoding of convolutional codes, as well as the performance analysis of convolutional codes over AWGN and Rayleigh fading channels. In this chapter, punctured convolutional codes, tail-biting convolutional codes, and their performance analysis are also discussed. Chapter 6 provides a treatment of the design of turbo codes, BCJR algorithm for iterative decoding of turbo codes, and performance analysis of turbo codes. In this chapter, enhanced turbo codes, enhanced list turbo decoding, and their performance evaluation are also described.

Chapter 7 focuses on the design and analysis of trellis-coded modulation schemes using both the conventional and turbo codes. Chapter 8 describes the design of low parity check codes (LDPC), quasi-cyclic (QC)-LDPC codes, decoding algorithms, and performance analysis of LDPC and QC-LDPC codes. The erasure correcting codes like Luby transform (LT) codes and Raptor codes are described in Chap. 9. The design of polar encoder and successive cancellation decoding (SCD), successive cancellation list decoding (SCLD), and multiple bit decision successive cancellation list decoding algorithms and their performance evaluation are provided in Chap. 10.

Chapter 11 provides an in-depth study of multiple-input multiple-output (MIMO) systems in which multiple antennas are used both at the transmitter and at the receiver. The advanced techniques for MIMO OFDM channel estimation are also described in this chapter. The design of space-time codes and implementations of MIMO systems are discussed in Chap. 12. Chapter 13 deals with the evolution of channel codes for 5G wireless communications.

The motivation in writing this book is to include modern topics of increasing importance such as turbo codes, LDPC codes, polar codes, LT and Raptor codes, and space-time coding in detail, in addition to the traditional RS codes and convolutional codes, and also to provide a comprehensive exposition of all aspects of coding for wireless channels. The text is integrated with MATLAB-based programs to enhance the understanding of the underlying theories of the subject.

This book is written at a level suitable for undergraduate and master students in electronics and communication engineering, electrical and computer engineering, computer science, and applied physics as well as for self-study by researchers, practicing engineers, and scientists. Depending on the chapters chosen, this text can be used for teaching a one- or two-semester course on coding for wireless channels. The prerequisite knowledge of the readers in principles of digital communication is expected.

Hyderabad, India

K. Deergha Rao

Contents

1	Introduction	1
1.1	Digital Communication System	1
1.2	Wireless Communication Channels	2
1.2.1	Binary Erasure Channel (BEC)	2
1.2.2	Binary Symmetric Channel (BSC)	3
1.2.3	Additive White Gaussian Noise Channel	3
1.2.4	Gilbert–Elliott Channel	4
1.2.5	Fading Channel	5
1.2.6	Fading	5
1.3	Statistical Models for Fading Channels	8
1.3.1	Probability Density Function of Rician Fading Channel	8
1.3.2	Probability Density Function of Rayleigh Fading Channel	9
1.3.3	Probability Density Function of Nakagami Fading Channel	9
1.4	Channel Capacity	10
1.4.1	Channel Capacity of Binary Erasure Channel	11
1.4.2	Channel Capacity of Binary Symmetric Channel	11
1.4.3	Capacity of AWGN Channel	11
1.4.4	Channel Capacity of Gilbert–Elliott Channels	13
1.4.5	Ergodic Capacity of Fading Channels	13
1.4.6	Outage Probability of a Fading Channel	15
1.4.7	Outage Capacity of Fading Channels	16
1.4.8	Capacity of Fading Channels with CSI at the Transmitter and Receiver	17
1.5	Channel Coding for Improving the Performance of Communication System	17
1.5.1	Shannon’s Noisy Channel Coding Theorem	17

1.5.2	Channel Coding Principle	18
1.5.3	Channel Coding Gain	18
1.6	Some Application Examples of Channel Coding	19
1.6.1	Error Correction Coding in GSM	19
1.6.2	Error Correction Coding in W-CDMA	20
1.6.3	Digital Video Broadcasting Channel Coding	20
1.6.4	Error Correction Coding in GPS L5 Signal	20
	References	21
2	Overview of the Performance of Digital Communication Over Fading Channels	23
2.1	BER Performance of Different Modulation Schemes in AWGN, Rayleigh, and Rician Fading Channels	23
2.1.1	BER of BPSK Modulation in AWGN Channel	24
2.1.2	BER of BPSK Modulation in Rayleigh Fading Channel	25
2.1.3	BER of BPSK Modulation in Rician Fading Channel	26
2.1.4	BER Performance of BFSK in AWGN, Rayleigh, and Rician Fading Channels	27
2.1.5	Comparison of BER Performance of BPSK, QPSK, and 16-QAM in AWGN and Rayleigh Fading Channels	29
2.2	Wireless Communication Techniques	30
2.2.1	DS-CDMA	31
2.2.2	FH-CDMA	34
2.2.3	OFDM	37
2.2.4	MC-CDMA	41
2.3	Diversity Reception	44
2.3.1	Receive Diversity with N Receive Antennas in AWGN	45
2.4	Diversity Combining Techniques	46
2.4.1	Selection Diversity	46
2.4.2	Equal Gain Combining (EGC)	47
2.4.3	Maximum Ratio Combining (MRC)	48
2.5	Problems	52
2.6	MATLAB Exercises	52
	References	53
3	Galois Field Theory	55
3.1	Set	55
3.2	Group	55
3.3	Field	56
3.4	Vector Spaces	57

3.5	Elementary Properties of Galois Fields	59
3.6	Galois Field Arithmetic	59
3.6.1	Addition and Subtraction of Polynomials	59
3.6.2	Multiplication of Polynomials	60
3.6.3	Multiplication of Polynomials Using MATLAB	60
3.6.4	Division of Polynomials	60
3.6.5	Division of Polynomials Using MATLAB	61
3.7	Polynomials Over Galois Fields	62
3.7.1	Irreducible Polynomial	62
3.7.2	Primitive Polynomials	63
3.7.3	Checking of Polynomials for Primitiveness Using MATLAB	63
3.7.4	Generation of Primitive Polynomials Using MATLAB	64
3.8	Construction of Galois Field $GF(2^m)$ from $GF(2)$	64
3.8.1	Construction of $GF(2^m)$, Using MATLAB	70
3.9	Minimal Polynomials and Conjugacy Classes of $GF(2^m)$	70
3.9.1	Minimal Polynomials	70
3.9.2	Conjugates of GF Elements	73
3.9.3	Properties of Minimal Polynomial	73
3.9.4	Construction of Minimal Polynomials	74
3.9.5	Construction of Conjugacy Classes Using MATLAB	76
3.9.6	Construction of Minimal Polynomials Using MATLAB	76
3.10	Problems	77
4	Linear Block Codes	79
4.1	Block Codes	79
4.2	Linear Block Codes	81
4.2.1	Linear Block Code Properties	81
4.2.2	Generator and Parity Check Matrices	82
4.2.3	Weight Distribution of Linear Block Codes	84
4.2.4	Hamming Codes	85
4.2.5	Syndrome Table Decoding	87
4.2.6	Hamming Codes Decoding	88
4.3	Cyclic Codes	89
4.3.1	The Basic Properties of Cyclic Codes	90
4.3.2	Encoding Algorithm for an (n, k) Cyclic Codes	91
4.3.3	Encoder for Cyclic Codes Using Shift Registers	93
4.3.4	Shift Register Encoders for Cyclic Codes	94
4.3.5	Cyclic Redundancy Check Codes	96

4.4	BCH Codes	98
4.4.1	BCH Code Design	98
4.4.2	Berlekamp's Algorithm for Binary BCH Codes Decoding	104
4.4.3	Chien Search Algorithm	105
4.5	Reed–Solomon Codes	109
4.5.1	Reed–Solomon Encoder	110
4.5.2	Decoding of Reed–Solomon Codes	112
4.5.3	Binary Erasure Decoding	123
4.5.4	Non-binary Erasure Decoding	123
4.6	Performance Analysis of RS Codes	127
4.6.1	BER Performance of RS Codes for BPSK Modulation in AWGN and Rayleigh Fading Channels	127
4.6.2	BER Performance of RS Codes for Non-coherent BFSK Modulation in AWGN and Rayleigh Fading Channels	131
4.7	Problems	133
4.8	MATLAB Exercises	134
	References	135
5	Convolutional Codes	137
5.1	Structure of Non-systematic Convolutional Encoder	137
5.1.1	Impulse Response of Convolutional Codes	139
5.1.2	Constraint Length	141
5.1.3	Convolutional Encoding Using MATLAB	141
5.2	Structure of Systematic Convolutional Encoder	142
5.3	The Structural Properties of Convolutional Codes	142
5.3.1	State Diagram	142
5.3.2	Catastrophic Convolutional Codes	142
5.3.3	Transfer Function of a Convolutional Encoder	144
5.3.4	Distance Properties of Convolutional Codes	148
5.3.5	Trellis Diagram	148
5.4	Punctured Convolutional Codes	154
5.5	The Viterbi Decoding Algorithm	155
5.5.1	Hard Decision Decoding	156
5.5.2	Soft Decision Decoding	158
5.6	Performance Analysis of Convolutional Codes	163
5.6.1	Binary Symmetric Channel	163
5.6.2	AWGN Channel	165
5.6.3	Rayleigh Fading Channel	167
5.7	Tail-Biting Convolutional Code	169
5.7.1	Tail-Biting Encoding	170

5.7.2	Tail-Biting Encoding Using MATLAB	171
5.7.3	Tail-Biting Decoding	172
5.8	Performance Analysis of Tail-Biting Convolutional Codes	173
5.9	Problems	177
5.10	MATLAB Exercises	179
	References	179
6	Turbo Codes	181
6.1	Non-recursive and Recursive Systematic Convolutional Encoders	181
6.1.1	Recursive Systematic Convolutional (RSC) Encoder	182
6.2	Turbo Encoder	183
6.2.1	Different Types of Interleavers	184
6.2.2	Turbo Coding Illustration	185
6.2.3	Turbo Coding Using MATLAB	189
6.2.4	Encoding Tail-Biting Codes with RSC (Feedback) Encoders	196
6.3	Turbo Decoder	199
6.3.1	The BCJR Algorithm	201
6.3.2	Turbo Decoding Illustration	205
6.3.3	Convergence Behavior of the Turbo Codes	213
6.3.4	EXIT Analysis of Turbo Codes	214
6.4	Performance Analysis of the Turbo Codes	217
6.4.1	Upper Bound for the Turbo Codes in AWGN Channel	217
6.4.2	Upper Bound for Turbo Codes in Rayleigh Fading Channel	218
6.4.3	Effect of Free Distance on the Performance of the Turbo Codes	222
6.4.4	Effect of Number of Iterations on the Performance of the Turbo Codes	224
6.4.5	Effect of Puncturing on the Performance of the Turbo Codes	225
6.5	Enhanced Turbo Codes	225
6.5.1	Enhanced Turbo Encoder	226
6.5.2	Enhanced List Turbo Decoder	226
6.6	Performance Analysis of Enhanced Turbo Codes	226
6.6.1	Performance of Enhanced Tail-Biting Turbo Codes Over AWGN and Rayleigh Fading Channels	226
6.6.2	Performance of Enhanced Turbo Codes with Tail-Biting List Decoding in AWGN Channels	228

6.7	Problems	229
6.8	MATLAB Exercises	230
	References	230
7	Bandwidth Efficient Coded Modulation	233
7.1	Set Partitioning	234
7.2	Design of the TCM Scheme	235
7.3	Decoding TCM	241
7.4	TCM Performance Analysis	241
7.4.1	Asymptotic Coding Gain	241
7.4.2	Bit Error Rate	242
7.4.3	Simulation of the BER Performance of a 8-State 8-PSK TCM in the AWGN and Rayleigh Fading Channels Using MATLAB	250
7.5	Turbo Trellis-Coded Modulation (TTCM)	254
7.5.1	TTCM Encoder	254
7.5.2	TTCM Decoder	255
7.5.3	Simulation of the BER Performance of the 8-State 8-PSK TTCM in AWGN and Rayleigh Fading Channels	257
7.6	Bit-Interleaved Coded Modulation	258
7.6.1	BICM Encoder	259
7.6.2	BICM Decoder	262
7.7	Bit-Interleaved Coded Modulation Using Iterative Decoding	263
7.7.1	BICM-ID Encoder and Decoder	263
7.7.2	Simulation of the BER Performance of 8-State 8-PSK BICM and BICM-ID in AWGN and Rayleigh Fading Channels	264
7.8	Problems	265
Appendix A	267
	References	272
8	Low Density Parity Check Codes	273
8.1	LDPC Code Properties	273
8.2	Construction of Parity Check Matrix H	274
8.2.1	Gallager Method for Random Construction of H for Regular Codes	274
8.2.2	Algebraic Construction of H for Regular Codes	275
8.2.3	Random Construction of H for Irregular Codes	276
8.3	Representation of Parity Check Matrix Using Tanner Graphs	277
8.3.1	Cycles of Tanner Graph	278
8.3.2	Detection and Removal of Girth 4 of a Parity Check Matrix	279

8.4	LDPC Encoding	282
8.4.1	Preprocessing Method.	282
8.5	Efficient Encoding of LDPC Codes	288
8.5.1	Efficient Encoding of LDPC Codes Using MATLAB	291
8.6	LDPC Decoding	292
8.6.1	LDPC Decoding on Binary Erasure Channel Using Message Passing Algorithm	292
8.6.2	LDPC Decoding on Binary Erasure Channel Using MATLAB	294
8.6.3	Bit-Flipping Decoding Algorithm	297
8.6.4	Bit-Flipping Decoding Using MATLAB	299
8.7	Sum Product Decoding	301
8.7.1	Log Domain Sum-Product Algorithm (SPA)	305
8.7.2	The Min-Sum Algorithm	306
8.7.3	Sum Product and Min-Sum Algorithms for Decoding of Rate 1/2 LDPC Codes Using MATLAB	309
8.8	EXIT Analysis of LDPC Codes	312
8.8.1	Degree Distribution	312
8.8.2	Ensemble Decoding Thresholds	313
8.8.3	EXIT Charts for Irregular LDPC Codes in Binary Input AWGN Channels	314
8.9	Performance Analysis of LDPC Codes	316
8.9.1	Performance Comparison of Sum-Product and Min-Sum Algorithms for Decoding of Regular LDPC Codes in AWGN Channel	316
8.9.2	BER Performance Comparison of Regular and Irregular LDPC Codes in AWGN Channel	317
8.9.3	Effect of Block Length on the BER Performance of LDPC Codes in AWGN Channel	317
8.9.4	Error Floor Comparison of Irregular LDPC Codes of Different Degree Distribution in AWGN Channel	319
8.10	Quasi Cyclic (QC)-LDPC CODES	320
8.10.1	Brief Description of QC-LDPC Codes	320
8.10.2	Base Matrix and Expansion	321
8.10.3	Performance Analysis of QC-LDPC Codes Over AWGN Channel	322
8.11	Problems	326
8.12	MATLAB Exercises	328
	References	328

9	LT and Raptor Codes	331
9.1	LT Codes Design	331
9.1.1	LT Degree Distributions	332
9.1.2	Important Properties of the Robust Soliton Distribution	333
9.1.3	LT Encoder	334
9.1.4	Tanner Graph of LT Codes	335
9.1.5	LT Decoding with Hard Decision	335
9.1.6	Hard Decision LT Decoding Using MATLAB	338
9.1.7	BER Performance of LT Decoding Over BEC Using MATLAB	339
9.2	Systematic LT Codes	341
9.2.1	Systematic LT Codes Decoding	342
9.2.2	BER Performance Analysis of Systematic LT Codes Using MATLAB	342
9.3	Raptor Codes	346
9.4	Problems	348
9.5	MATLAB Exercises	348
	References	348
10	Polar Codes	351
10.1	Channel Polarization	351
10.1.1	Channel Combining Phase	352
10.1.2	Channel Splitting Phase	357
10.1.3	Polarization of Binary Erasure Channels	358
10.1.4	Polarization Theorem	361
10.1.5	Polarization of AWGN Channels	362
10.2	Polar Encoder Structures and Encoding	362
10.2.1	Polar Encoder for $N = 2$	363
10.2.2	Polar Encoder for $N = 4$	363
10.2.3	Polar Encoder for $N = 4$ with Input Data Vector Permutated	364
10.2.4	Polar Encoder for $N = 8$	364
10.2.5	Polar Encoder for $N = 8$ with Input Data Vector Permutated	365
10.2.6	Non-systematic Polar Encoding Using MATLAB	367
10.2.7	Systematic Polar Encoding	368
10.2.8	Efficient Systematic Polar Encoding Algorithm	369
10.3	Polar Decoding	371
10.3.1	Successive Cancellation Decoding	371
10.3.2	SC Decoding Algorithm	374
10.3.3	Successive Cancellation List Decoding	381

10.4	Problems	381
10.5	MATLAB Exercises	382
	References	383
11	MIMO System	385
11.1	What Is MIMO?	385
11.2	MIMO Channel Model	385
11.2.1	The Frequency Flat MIMO Channel	385
11.2.2	The Frequency-Selective MIMO Channel	387
11.2.3	MIMO-OFDM System	387
11.3	Channel Estimation	388
11.3.1	LS Channel Estimation	389
11.3.2	DFT-Based Channel Estimation	389
11.3.3	MIMO-OFDM Channel Estimation Using LS	390
11.3.4	Channel Estimation Using MATLAB	390
11.4	MIMO Channel Decomposition	392
11.5	MIMO Channel Capacity	399
11.5.1	Capacity of Deterministic MIMO Channel When CSI Is Known to the Transmitter	399
11.5.2	Deterministic MIMO Channel Capacity When CSI Is Unknown at the Transmitter	401
11.5.3	Random MIMO Channel Capacity	403
11.6	MIMO-OFDM Channel Estimation Using OMP Algorithm	411
11.6.1	OMP Algorithm	412
11.7	MIMO Channel Equalization	416
11.7.1	Zero Forcing (ZF) Equalization	417
11.7.2	Minimum Mean Square Error (MMSE) Equalization	418
11.8	Problems	419
11.9	MATLAB Exercises	420
	References	421
12	Space-Time Coding	423
12.1	Space-Time-Coded MIMO System	423
12.2	Space-Time Block Code (STBC)	424
12.2.1	Rate Limit	424
12.2.2	Orthogonality	425
12.2.3	Diversity Criterion	425
12.2.4	Performance Criteria	426
12.2.5	Decoding STBCs	427
12.3	Alamouti Code	427
12.3.1	2-Transmit, 1-Receive Alamouti STBC Coding	428
12.3.2	2-Transmit, 2-Receive Alamouti STBC Coding	429
12.3.3	Theoretical BER Performance of BPSK Alamouti Codes Using MATLAB	433

12.4	Higher-Order STBCs	434
12.4.1	3-Transmit, 4-Receive STBC Coding	435
12.4.2	Simulation of BER Performance of STBCs Using MATLAB	439
12.5	Space-Time Trellis Coding	442
12.5.1	Space-Time Trellis Encoder	443
12.5.2	Simulation of BER Performance of 4-state QPSK STTC Using MATLAB	451
12.6	MIMO-OFDM Implementation	457
12.6.1	Space-Time-Coded OFDM	459
12.6.2	Space-Frequency Coded OFDM	459
12.6.3	Space-Time-Frequency Coded OFDM	460
12.7	Problems	461
12.8	MATLAB Exercises	463
	References	463
13	Channel Codes Evolution for 5G	465
13.1	5G Requirements	465
13.2	QC-LDPC and Polar Codes for eMBB	466
13.2.1	Performance Evaluation of QC-LDPC Codes for eMBB Data Channel	467
13.2.2	Performance Evaluation of Polar Codes for eMBB Control Channel	468
13.3	Evaluation of Enhanced Turbo Codes and Polar Codes for URLLC	470
13.3.1	Decoding Latency	473
13.3.2	Decoding Complexity	474
13.4	Channel Codes for mMTC	474
	References	476

About the Author

K. Deergha Rao is Professor at the Department of Electronics and Communication Engineering, Vasavi College of Engineering, Osmania University, Hyderabad, India. He is former Director and Professor of Research and Training Unit for Navigational Electronics (NERTU), Osmania University. He was a postdoctoral fellow and part-time professor for four years at the Department of Electrical and Computer Engineering, Concordia University, Montreal, Canada. His teaching areas are signals and systems, digital signal processing, channel coding techniques, and MIMO communication systems. Professor Rao has executed research projects for premium Indian organizations such as DRDO, HAL, and BEL and is interested in research areas of wireless channel coding, MIMO-OFDM communications, image processing, cryptosystems, and VLSI signal processing. He has also served as the founder chairman of the joint chapter of IEEE Communications Society and IEEE Signal Processing Society in Hyderabad from 2010–2012 and communications track chair for IEEE INDICON 2011 held in Hyderabad. Five students have been so far awarded Ph.D. degrees under Professor Rao, while three are currently working towards their Ph.D.

An awardee of the IETE K. S. Krishnan Memorial Award for the best system oriented paper in 2013, Prof. Rao has presented papers at IEEE International conferences several times in the USA, Switzerland, Russia, and Thailand. He has more than 100 publications to his credit including more than 60 publications in the IEEE journals and conference proceedings. Professor Rao is the author of two books—*Channel Coding Techniques for Wireless Communications* (Springer, 2015) and *Signals and Systems* (Springer, 2018)—and has co-authored *Digital Signal Processing* (Jaico Publishing House, 2012) and *Digital Signal Processing: Theory and Practice* (Springer, 2018). He is an editorial board member for the *International Journal of Sustainable Aviation*.

Chapter 1

Introduction



In this chapter, a digital communication system with coding is first described. Second, various wireless communication channels, their probability density functions, and capacities are discussed. Further, Shannon's noisy channel coding theorem, channel coding principle and channel coding gain are explained. Finally, some application examples of channel coding are included.

1.1 Digital Communication System

A communication system is a means of conveying information from one user to another user. The digital communication system is one in which the data is transmitted in digital form. A digital communication system schematic diagram is shown in Fig. 1.1. The source coding is used to remove redundancy from source information for efficient transmission. The transmitted signal power and channel bandwidth are the key parameters in the design of digital communication system. Using these parameters, the signal energy per bit (E_b) to noise power spectral density (N_0) ratio is determined. This ratio is unique in determining the probability of bit error, often referred to as bit error rate (BER). In practice, for a fixed E_b/N_0 , acceptable BER is possible with channel coding. This can be achieved by adding additional digits to the transmitted information stream. These additional digits do not have any new information, but they make it possible for the receiver to detect and correct errors, thereby reducing the overall probability of error.

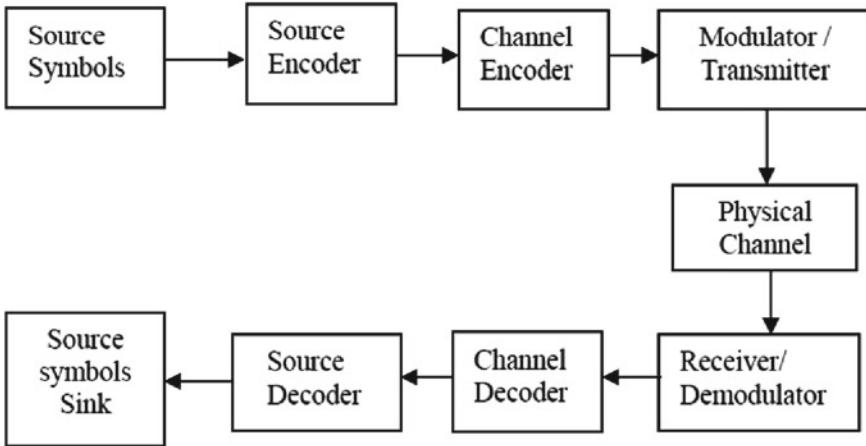


Fig. 1.1 Digital communication system with coding

1.2 Wireless Communication Channels

1.2.1 *Binary Erasure Channel (BEC)*

Erasure is a special type of error with known location. The binary erasure channel (BEC) transmits one of the two binary bits 0 and 1. However, an erasure “ e ” is produced when the receiver receives an unreliable bit. The BEC output consists of 0, 1, and e as shown in Fig. 1.2. The BEC erases a bit with probability ε , called the erasure probability of the channel. Thus, the channel transition probabilities for the BEC are:

Fig. 1.2 Binary erasure channel

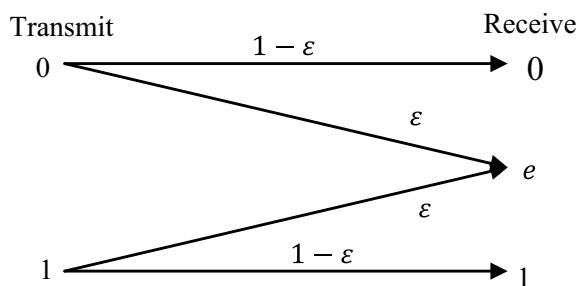
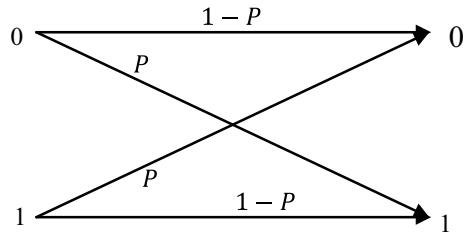


Fig. 1.3 Binary symmetric channel



$$\left. \begin{array}{l} P(y = 0|x = 0) = 1 - \varepsilon, \\ P(y = e|x = 0) = \varepsilon, \\ P(y = 1|x = 0) = 0, \\ P(y = 0|x = 1) = 0, \\ P(y = e|x = 1) = \varepsilon, \\ P(y = 1|x = 1) = 1 - \varepsilon. \end{array} \right\} \quad (1.1)$$

1.2.2 Binary Symmetric Channel (BSC)

The binary symmetric channel (BSC) is a discrete memoryless channel that has binary symbols both in the input and output. It is symmetric because the probability for receiving 0 when 1 is transmitted is same as the probability for receiving 1 when 0 is transmitted. This probability is called the crossover probability of the channel denoted by P as shown in Fig. 1.3. The probability for no error, i.e., receiving the same as transmitted is $1 - P$. Hence, the channel transition probabilities for the BSC are:

$$\left. \begin{array}{l} P(y = 0|x = 0) = 1 - P, \\ P(y = 0|x = 1) = P, \\ P(y = 1|x = 0) = P, \\ P(y = 1|x = 1) = 1 - P, \end{array} \right\} \quad (1.2)$$

1.2.3 Additive White Gaussian Noise Channel

In an AWGN channel, the signal is degraded by white noise η which has a constant spectral density and a Gaussian distribution of amplitude. The Gaussian distribution has a Probability Density Function (PDF) given by

$$P_{df}(\eta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\eta^2}{2\sigma^2}\right) \quad (1.3)$$

where σ^2 is the variance of a Gaussian random process.

1.2.4 Gilbert–Elliott Channel

For bursty wireless channels, the Gilbert–Elliott (GE) channel [1, 2] is one of the simplest and practical models. The GE channel is a discrete-time stationary model as shown in Fig. 1.4 with two states: one bad state or burst state “2” wherein a BSC resides with high-error probabilities ($1 - P_2$) and the other state is a good state “1” wherein a BSC resides with low-error probabilities ($1 - P_1$).

Another common GE Example is that the BEC resides in a bad state with ε close to unity and assigns erasures to all of the bits transmitted during the high-error-rate (bad) state.

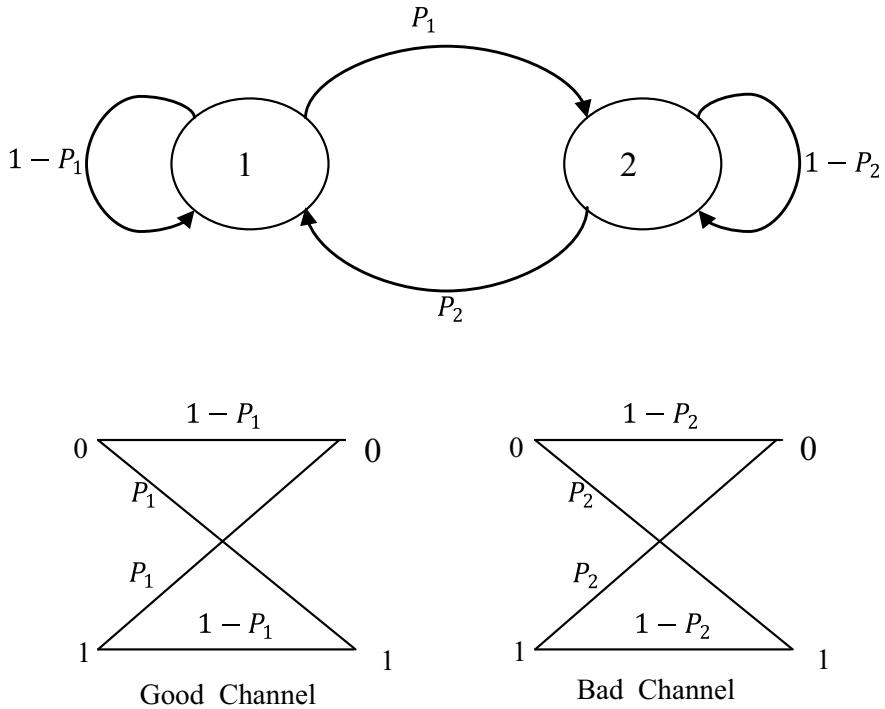


Fig. 1.4 Two-state channel

1.2.5 Fading Channel

In the radio channel, the received power is affected by the attenuations due to the combinations of the following effects

1. *The Path loss:* It is the signal attenuation. The power received by the receiving antenna decreases when the distance between transmitter and receiver increases. The power attenuation is proportional to (distance) $^\alpha$, where α values range from 2 to 4. When the distance varies with time, the path loss also varies.
2. *The Shadowing loss:* It is due to the absorption of the radiated signal by scattering structure. It is derived from a random variable with log-normal distribution.
3. *The Fading loss:* The combination of multipath propagation and the Doppler frequency shift produces the random fluctuations in the received power which gives the fading losses.

1.2.6 Fading

Fading gives the variations of the received power along with the time. It is due to the combination of multipath propagation and the Doppler frequency shift which gives the time-varying attenuations and delays that may degrade the communication system performance. The received signal is a distorted version of the transmitted signal which is a sum of the signal components from the various paths with different delays due to multipath and motion.

Let T_s be the duration of a transmitted signal and B_x be the signal bandwidth. The fading channel can be classified based on coherence time and coherence bandwidth of the channel. The coherence time and coherence bandwidth of a channel are defined as

Doppler spread:

Doppler shift is defined as the perceived change in the frequency of the electromagnetic wave due to the mobility of the mobile user.

If the mobile station is moving with a velocity v at an angle θ with the line joining mobile user and base station, the Doppler shift f_d is expressed by

$$f_d = \left(\frac{v}{c} \cos \theta \right) f_c \quad (1.4a)$$

where f_c is the carrier frequency, and c is the velocity of light.

From the above equation, it can be observed that the Doppler shift is maximum when $\theta = 0$ and the Doppler shift is zero when $\theta = 0 = \frac{\pi}{2}$ that is, when the mobile motion is perpendicular to the receive direction.

Thus, the maximum Doppler shift $f_{d\max}$ is given by

$$f_{d\max} = \left(\frac{v}{c}\right)f_c \quad (1.4b)$$

Then, the delay spread $B_D = 2f_{d\max}$

The coherence time of the channel T_c is

$$T_c \triangleq \frac{1}{2B_D} = \frac{1}{4f_{d\max}} \quad (1.4c)$$

The coherence time implies that the channel changes at every coherence time duration, and the channel is to be estimated at least once in every coherence time interval.

Example 1.1 Consider a mobile user is moving at 40 kmph with $f_c = 2.4$ GHz.

Compute the coherence time.

Solution

$$\begin{aligned} 40 \text{ kmph} &= 40 \times \frac{5}{18} = \frac{2000}{18} = 11.1111 \text{ m/s} \\ f_{d\max} &= \frac{11.1111}{3 \times 10^8} \times 2400 \times 10^6 = 88.8888 \text{ Hz} \\ T_c &= \frac{1}{4f_{d\max}} = \frac{1}{4 \times 88.8888} = 22.5 \text{ ms} \end{aligned}$$

Delay spread:

The maximum among the path delay differences, a significant change occurs when the frequency change exceeds the inverse of T_D , called the *delay spread* of the channel.

The channel bandwidth of the channel B_c is

$$B_c \triangleq \frac{1}{T_D} \quad (1.5)$$

For example, if a 4-multipath channel with the delays corresponding to the first and the last arriving are 0 μ s and 6 μ s, the maximum delay spread

$$T_D = 6 \mu\text{s} - 0 \mu\text{s} = 6 \mu\text{s}$$

RMS Delay spread:

In a typical wireless channel, the later arriving paths are with lower power due to larger propagation distances and weaker reflections. In such a scenario, maximum delay spread metric is not reliable.

The RMS delay spread is defined by

$$T_{\text{DRMS}} = \sqrt{\frac{\sum_{i=0}^{L-1} g_i (\tau_i - \tilde{\tau})^2}{\sum_{i=0}^{L-1} g_i}}$$

where

τ_i is the delay of the i th path.

g_i is the power corresponding to the i th path.

$\tilde{\tau}$ is the average delay given by

$$\tilde{\tau} = \frac{\sum_{i=0}^{L-1} g_i \tau_i}{\sum_{i=0}^{L-1} g_i}$$

Example 1.2 Consider a wireless channel consists of $L = 4$ multipath with the delays and power tabulated as follows

i	τ_i (μs)	g_i
0	0	1
1	1	0.1
2	3	0.1
3	5	0.01

Solution

$$\tilde{\tau} = \frac{1 \times 0 + 0.1 \times 1 + 0.1 \times 3 + 0.01 \times 5}{1 + 0.1 + 0.1 + 0.01} \mu s = 0.3719 \mu s$$

$$T_{\text{DRMS}} = \sqrt{\frac{1 \times (0 - 0.3719)^2 + 0.1 \times (1 - 0.3719)^2 + 0.1 \times (3 - 0.3719)^2 + 0.01 \times (5 - 0.3719)^2}{1 + 0.1 + 0.1 + 0.01}} \mu s \\ = 0.9459 \mu s$$

1.2.6.1 Fading Channels Classification

The classification of fading channels is shown in Fig. 1.5.

The fast fading causes short burst errors which are easy to correct. The slow fading will affect many successive symbols leading to long burst errors. Due to energy absorption and scattering in physical channel propagation media, the transmitted signal is attenuated and becomes noisy. The attenuation will vary in mobile

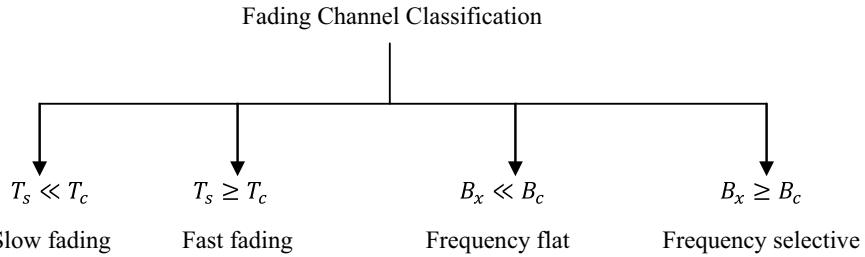


Fig. 1.5 Classification of fading channels

communications based on the vehicle speed, surrounding trees, buildings, mountains, and terrain. Based on the receiver location, moving receiver signals interfere with one another and take several different paths. As such, the wireless channels are called multipath fading channels. Hence, the additive white Gaussian noise (AWGN) assumption for wireless channels is not realistic. Thus, the amplitudes in the wireless channels are often modeled using Rayleigh or Rician probability density function.

The most common fading channel models are

1. Flat independent fading channel
2. Block fading channel.

In flat independent fading channel, the attenuation remains constant for one symbol period and varies from symbol to symbol. Whereas in block fading channel, the attenuation is constant over a block of symbols and varies from block to block.

1.3 Statistical Models for Fading Channels

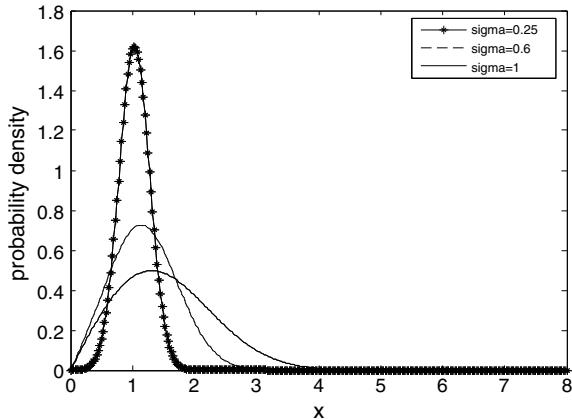
1.3.1 Probability Density Function of Rician Fading Channel

When the received signal is made up of multiple reflective rays plus a significant line-of-sight (non-faded) component, the received envelope amplitude has a Rician Probability Density Function (PDF) as given in Eq. (1.6), and the fading is referred to as Rician fading.

$$\begin{aligned}
 P_{df}(x) &= \frac{x}{\sigma^2} \exp\left(-\frac{(x^2+A^2)}{2\sigma^2}\right) I_0\left(\frac{xA}{\sigma^2}\right); \text{ for } x \geq 0, A \geq 0 \\
 &= 0 \quad ; \text{ Otherwise}
 \end{aligned} \tag{1.6}$$

where x is the amplitude of the received faded signal, I_0 is the zero-order modified Bessel function of the first kind, and A denotes the peak magnitude of the non-faded signal component called the specular component. The Rician PDF for different values of sigma and $A = 1$ is shown in Fig. 1.6.

Fig. 1.6 Probability density of Rician fading channel



1.3.2 Probability Density Function of Rayleigh Fading Channel

Rayleigh fading occurs when there are multiple indirect paths between the transmitter and the receiver and no direct non-fading or line-of-sight (LOS) path. It represents the worst case scenario for the transmission channel. Rayleigh fading assumes that a received multipath signal consists of a large number of reflected waves with independent and identically distributed phase and amplitude. The envelope of the received carrier signal is Rayleigh distributed in wireless communications [3].

As the magnitude of the specular component approaches zero, the Rician PDF approaches a Rayleigh PDF expressed as follows:

$$P_{df}(x) = \begin{cases} \frac{x}{\sigma^2} \exp\left(-\frac{x^2}{2\sigma^2}\right) & \text{for } x \geq 0 \\ 0 & \text{Otherwise} \end{cases} \quad (1.7)$$

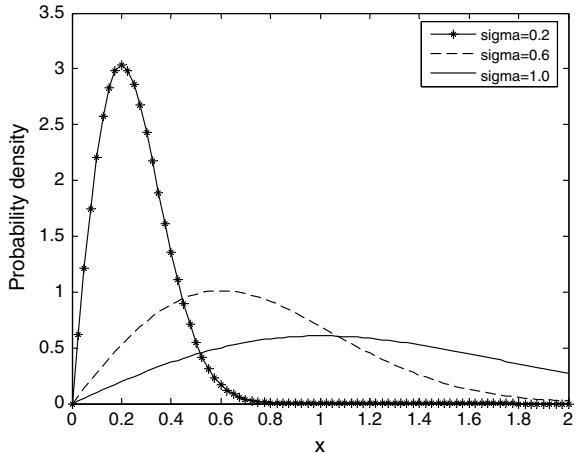
The Rayleigh PDF for different values of sigma is shown in Fig. 1.7.

Additive white Gaussian noise and Rician channels provide fairly good performance corresponding to an open country environment, while Rayleigh channel, which best describes the urban environment fading, provides relatively worse performance.

1.3.3 Probability Density Function of Nakagami Fading Channel

The Nakagami model is another very popular empirical fading model [4]

Fig. 1.7 Probability density of Rayleigh fading channel



$$P_{df}(r) = \frac{2}{\Gamma(m)} \left(\frac{m}{2\sigma^2} \right)^m r^{2m-1} e^{-m \frac{r^2}{2\sigma^2}} \quad (1.8)$$

where

$\sigma^2 = \frac{1}{2} E[r^2]$, $\Gamma(\cdot)$ is the gamma function.

$m \geq \frac{1}{2}$ is the fading figure.

The received instantaneous power r^2 satisfies a gamma distribution. The phase of the signal is uniformly distributed in $[0, 2\pi]$. The Nakagami distribution is a general model obtained from experimental data fitting and its shape very similar to that of the Rice distribution. The shape parameter “ m ” measures the severity of fading.

when

$m = 1$, it is Rayleigh fading.

$m \rightarrow \infty$, it is AWGN channel, that is, there is no fading.

$m > 1$, it is close to Rician fading.

However, due to lack of physical basis, the Nakagami distribution is not as popular as the Rician and Rayleigh fading models in mobile communications. Many other fading channel models are discussed in [5].

1.4 Channel Capacity

Channel capacity can be defined as the maximum rate at which the information can be transmitted over a reliable channel.

$$\text{Spectral or Bandwidth Efficiency} = \frac{\text{Transmission rate}}{\text{Channel Bandwidth}} = \frac{R_s \mathcal{H}}{B} \text{ bits/s/Hz}$$
(1.9)

where R_s is the symbol rate, \mathcal{H} is the entropy.

The channel capacity is also known as Shannon's capacity which can be defined as the average mutual information for a channel with energy constraint.

1.4.1 Channel Capacity of Binary Erasure Channel

The channel capacity of binary erasure channel is

$$C_{BEC} = 1 - \varepsilon$$
(1.10)

ε is the probability of a bit erasure, which is represented by the symbol e .

1.4.2 Channel Capacity of Binary Symmetric Channel

The channel capacity of binary symmetric channel is

$$C_{BSC} = 1 - \mathcal{H}(P)$$
(1.11)

$\mathcal{H}(P)$ is the binary entropy function given by Ryan and Lin [6]

$$\mathcal{H}(P) = -P \log_2(P) - (1 - P) \log_2(1 - P)$$
(1.12)

P is the probability of a bit error.

1.4.3 Capacity of AWGN Channel

An additive white Gaussian noise (AWGN) channel can be expressed by the following input–output relationship

$$y = x + \eta$$
(1.13)

where x is the transmitted source signal, y denotes the output of the channel, η is a real Gaussian process with zero mean, variance $\sigma_\eta^2 = E[\eta^2]$, and two-sided power spectral density $\frac{N_0}{2}$. The mutual information $I(x; y)$ with constraint on the energy of the input signal can be expressed as

$$I(x, y) = \mathcal{H}(y) - \mathcal{H}(\eta) \quad (1.14)$$

where

$\mathcal{H}(y)$ is the entropy of the channel output.

$\mathcal{H}(\eta)$ is the entropy of the AWGN.

Since $\sigma_y^2 = \sigma_x^2 + \sigma_\eta^2$, the entropy $\mathcal{H}(y)$ is bounded by $\frac{1}{2} \log_2 \pi e (\sigma_x^2 + \sigma_\eta^2)$ and thus

$$\begin{aligned} I(x, y) &\leq \frac{1}{2} \log_2 \pi e (\sigma_x^2 + \sigma_\eta^2) - \frac{1}{2} \log_2 \pi e \sigma_\eta^2 \\ &= \frac{1}{2} \log_2 \left(1 + \frac{\sigma_x^2}{\sigma_\eta^2} \right) \end{aligned} \quad (1.15)$$

The mutual information $I(x, y)$ is maximum when the input x is a real Gaussian process with zero mean and variance σ_x^2 . The capacity of the channel is the maximum information that can be transmitted from x to y by varying the probability density function P_{df} of the transmit signal x . The signal-to-noise ratio (SNR) is defined by

$$\text{SNR} \triangleq \frac{\sigma_x^2}{\sigma_\eta^2} \quad (1.16)$$

Thus, the capacity of an AWGN channel is given by

$$\mathbf{C} = \frac{1}{2} \log_2 (1 + \text{SNR}) \text{ bits/s/Hz} \quad (1.17)$$

Since $\sigma_x^2 = BE_s$ and $\sigma_\eta^2 = BN_0$, Eq. (1.17) can be rewritten as

$$\mathbf{C} = \frac{1}{2} \log_2 \left(1 + 2 \frac{E_s}{N_0} \right) \text{ bits/s/Hz} \quad (1.18)$$

where B is the bandwidth, E_s denotes the symbol energy, and N_0 represents the noise spectral density.

If x and η are independent complex Gaussian processes, the channel capacity can be expressed as

$$\mathbf{C} = \log_2 (1 + \text{SNR}) \text{ bits/s/Hz} \quad (1.19)$$

Since $\sigma_x^2 = BE_s$ and $\sigma_\eta^2 = BN_0$ for complex Gaussian process, Eq. (1.19) can be rewritten as

$$\mathbf{C} = \log_2 \left(1 + \frac{E_s}{N_0} \right) \text{ bits/s/Hz} \quad (1.20)$$

Example 1.3 What is the capacity of a channel with an SNR of 20 dB.

Solution $C = \log_2(1 + 20) = 6.65$ bits/s/Hz.

The capacity is increasing as a log function of the SNR, which is a slow increase. Clearly increasing the capacity by any significant factor takes an enormous amount of power.

1.4.4 Channel Capacity of Gilbert–Elliott Channels

The channel capacity of Gilbert–Elliott Channel is given by Ryan and Lin [6]

$$\mathbf{C}_{GE} = \sum_{s=1}^S P_s \mathbf{C}_s \quad (1.21)$$

where P_s is the probability of being state in s state, and \mathbf{C}_s is the capacity of the channel in s state.

1.4.5 Ergodic Capacity of Fading Channels

A slow flat fading channel with additive white Gaussian noise (AWGN) can be expressed by the following input–output relationship

$$y = hx + \eta \quad (1.22)$$

where x is the transmitted source signal, y denotes the output of the channel, η is the AWGN, and h is a Gaussian random variable with Rician or Rayleigh PDF.

The fading channel model given in Eq. (1.22) can be seen as a Gaussian channel with attenuation h . If h is assumed to be an ergodic process, the capacity of the fading channel is the Ergodic capacity computed by the following expression

$$\mathbf{C} = E[\log_2(1 + h^2 \text{SNR})] \text{ bits/s/Hz} \quad (1.23)$$

where the expectation $E[\cdot]$ is with respect to random variable h . If $E[h^2] = 1$, Eq. (1.23) is always less than AWGN channel capacity since $E[f(X)] \leq f(E[X])$ according to Jensen inequality. If h has Rayleigh PDF, computation of Eq. (1.24) yields [5]

$$\mathbf{C} = \log_2 e \cdot \exp\left(\frac{1}{\text{SNR}}\right) \cdot \text{expint}\left(\frac{1}{\text{SNR}}\right) \text{ bits/s/Hz} \quad (1.24)$$

where

$$\text{expint}(x) \triangleq \int_x^{\infty} \frac{e^t}{t} dt$$

which is the capacity of the independent Rayleigh fading channel with no constraint on the constellation of the input signal. The following MATLAB program is written and used to compute the AWGN channel capacity in AWGN and the ergodic capacity of a Rayleigh fading channel.

Program 1.1 MATLAB program to compute capacity of AWGN channel and ergodic capacity of Rayleigh fading channel with channel state information (CSI).

```
% capacity of AWGN channel and ergodic capacity of Rayleigh fading
%channel state information (CSI).
clear all
close all
SNRdB = [-10:0.1:30];
SNRlin = 10.^ (SNRdB/10);
C_AWGN = log2(1 + SNRlin); % AWGN
C_Rayleigh = log2(exp(1)) * exp(1 ./ SNRlin) .* expint(1 ./ SNRlin); % Rayleigh
plot(SNRdB, C_AWGN, '--', SNRdB, C_Rayleigh, '-');
xlabel('SNR(dB)'), ylabel('{it Capacity} (bit/s/Hz)');
legend('AWGN', 'Rayleigh fading');
```

The SNR versus capacity plot obtained from the above MATLAB program is shown in Fig. 1.8. From Fig. 1.8, it can be observed that there is a much lower performance difference between the capacities of AWGN and Rayleigh channels. This is highly indicative that the coding of fading channels will yield considerable coding gain for large SNR.

Example 1.4 For large SNR's, verify that the SNR required to obtain the same ergodic capacity for the AWGN channel and the independent Rayleigh fading channel differs by 2.5 dB.

Solution AWGN channel capacity is given by $C = \log_2(1 + \text{SNR})$ bits/s/Hz.

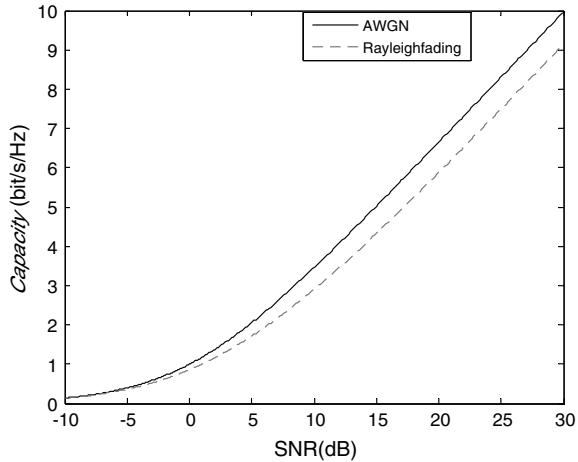
For large SNRs, the above equation can be approximated as follows:

$$C = \log_2(\text{SNR}) \text{ bits/s/Hz}$$

The ergodic capacity in Rayleigh fading channel is given by

$$C = \log_2 e \cdot \exp\left(\frac{1}{\text{SNR}}\right) \cdot \text{expint}\left(\frac{1}{\text{SNR}}\right)$$

Fig. 1.8 Capacities of AWGN and Rayleigh fading channel



For large SNRs, the above equation can be rewritten as

$$C_{\text{Rayleigh}} = \log_2(\text{SNR}) - 0.8327$$

Since the capacity of an AWGN channel for large SNRs can be approximated as $\log_2(\text{SNR})$, the above relation can be rewritten as follows:

$$C_{\text{Rayleigh}} = C_{\text{AWGN}} - 0.8327$$

Thus, the capacity for AWGN channel and the Rayleigh fading channel differs by 0.8327.

The difference in dB can be expressed as follows:

$$10 \log_{10}(2^{0.8327}) = 2.5 \text{ dB}$$

1.4.6 Outage Probability of a Fading Channel

A mobile user will experience rapid changes in SNR as fading channels lead to an oscillating SNR at different locations. As such, the channel can be characterized by an average SNR and BER can be computed by using this. If BER is below a threshold, then it is not the primary concern for many applications. A more meaningful measure is outage probability, which is the percentage of time that an acceptable quality of communication is not available.

The outage probability of a fading channel is the probability with which the information outage occurs when the transmission rate exceeds the capacity.

The outage probability for a Rayleigh fading channel with the same SNR as that of AWGN is given by Du and Swamy [3]

$$P_{\text{out}} = 1 - \exp\left(\frac{1 - 2^{C_{\text{out}}}}{\text{SNR}}\right) \quad (1.25)$$

1.4.7 Outage Capacity of Fading Channels

The outage capacity of a fading channel is the maximum rate supported by the channel for a given outage probability of the channel. The C_{out} can be expressed as follows:

$$C_{\text{out}} = \log_2(1 - \text{SNR} \log(1 - P_{\text{out}})) \quad (1.26)$$

The following MATLAB program is written and used to compute the outage capacity of Rayleigh fading channels for different outage probabilities.

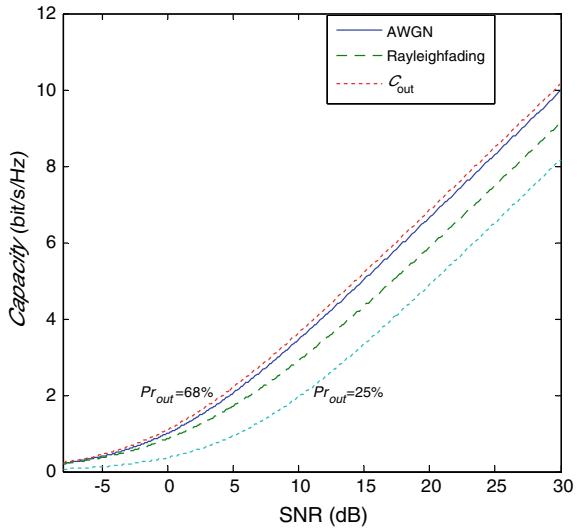
Program 1.2 MATLAB program to compute outage capacities of the Rayleigh fading channel

```
% outage capacities of Rayleigh fading channel
clear all
close all
SNRdB = [-10:0.1:30];
SNRlin = 10.^{SNRdB/10};
C_AWGN = log2(1 + SNRlin);% AWGN
C_Rayleigh = log2(exp(1)) * exp(1./ SNRlin) .* expint(1./ SNRlin);%
Rayleigh
P_out = 25e-2;
C_out_25 = log2(1 - SNRlin * log(1-P_out));
P_out = 68e-2;
C_out_68 = log2(1 - SNRlin * log(1-P_out));
plot(SNRdB, C_AWGN, '-', SNRdB, C_Rayleigh, '--', SNRdB,
C_out_68, ':', SNRdB, C_out_25, ':');
xlabel('SNRdB'), ylabel('Outage Capacity (bit/s/Hz)');
legend('AWGN', 'Rayleigh fading', 'C_{out}=4%', 'C_{out}=46%', 'C_{out}=64%');
```

The outage capacity of Rayleigh fading channel for different outage probabilities obtained from the above program is shown in Fig. 1.9.

It is observed from Fig. 1.9 that at $p_{\text{out}} = 68\%$, C_{out} is greater than the capacity of AWGN channel.

Fig. 1.9 Outage capacities of Rayleigh fading channel



1.4.8 Capacity of Fading Channels with CSI at the Transmitter and Receiver

The ergodic capacity of a Rayleigh fading channel with channel state information (CSI) at the transmitter and receiver is given by [7]

$$C = \int_{\gamma_0}^{\infty} B \log_2 \left(\frac{\gamma}{\gamma_0} \right) P_{df}(\gamma) d\gamma \text{ bits/s/Hz} \quad (1.27)$$

where

γ is the signal-to-noise ratio(SNR).

γ_0 is the cutoff SNR.

$P_{df}(\gamma)$ is the probability density function of γ due to the fading channel.

1.5 Channel Coding for Improving the Performance of Communication System

1.5.1 Shannon's Noisy Channel Coding Theorem

Any channel affected by noise possesses a specific “channel capacity” C a rate of conveying information that can never be exceeded without error, but in principle, an

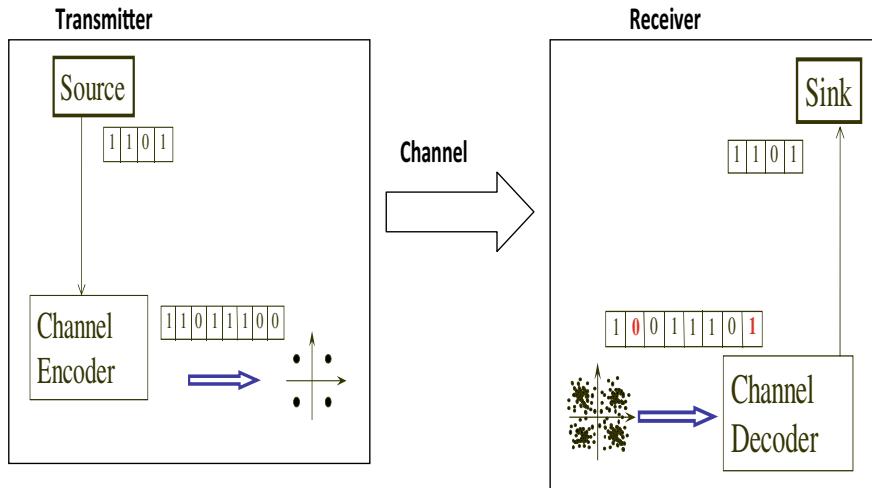


Fig. 1.10 Illustration of channel coding principle

error-correcting code always exists such that information can be transmitted at rates less than C with an arbitrarily low BER.

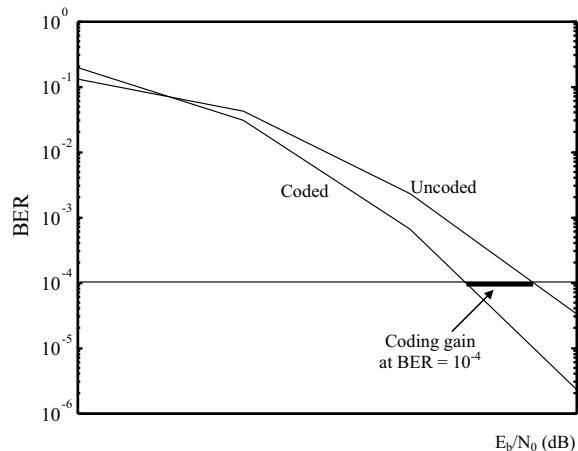
1.5.2 *Channel Coding Principle*

The channel coding principle is to add redundancy to minimize error rate as illustrated in Fig. 1.10.

1.5.3 *Channel Coding Gain*

The bit error rate (BER) is the probability that a binary digit transmitted from the source received erroneously by the user. For required BER, the difference between the powers required for without and with coding is called the coding gain. A typical plot of BER versus E_b/N_0 (bit energy to noise spectral density ratio) with and without channel coding is shown in Fig. 1.11. It can be seen that coding can arrive at the same value of the BER at lower E_b/N_0 than without coding. Thus, the channel coding yields coding gain which is usually measured in dB. Also, the coding gain usually increases with a decrease in BER.

Fig. 1.11 Illustration of coding gain



1.6 Some Application Examples of Channel Coding

1.6.1 Error Correction Coding in GSM

Each speech sample of 20 ms duration is encoded by RPE-LPC as 260 bits with total bit rate of 13 kbps. The 260 bits are classified into three types based on their sensitiveness as shown in Fig. 1.12.

The 50 bits in Type Ia are the most sensitive to bit errors, the next 132 bits in Type Ib are moderately sensitive to bit errors, and the other 78 bits in Type II do not need any protection. The Type Ia bits are encoded using a cyclic encoder. The Type Ib bits and the encoded Type Ia bits are encoded using convolutional encoder. The Type II bits are finally added to the convolutional encoder output bits.

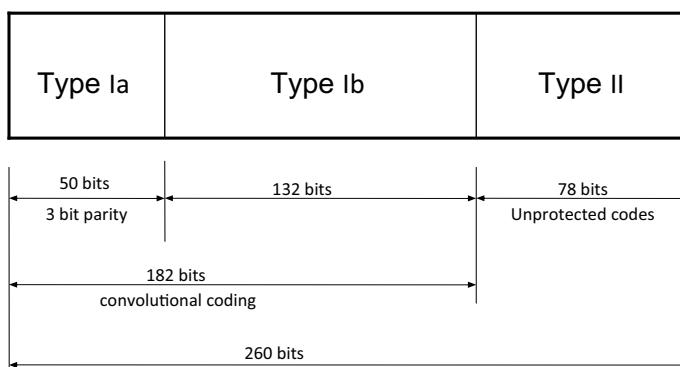


Fig. 1.12 Classification of speech sample 3 in GSM

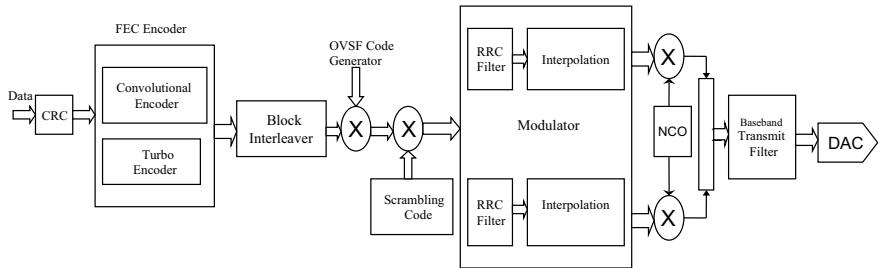


Fig. 1.13 Error correction coding in W-CDMA. CRC = cyclic redundancy check. DAC = digital to analog convertor. NCO = numerically controlled oscillator. OVSF = orthogonal variable spreading factor. RRC = root raised cosine

1.6.2 Error Correction Coding in W-CDMA

The W-CDMA standard has defined two error correction coding schemes as shown in Fig. 1.13 for different quality of services. The W-CDMA standard uses convolutional encoding for voice and MPEG4 applications and uses turbo encoding for data applications with longer time delays. The convolutional encoding gives a bit error rate (BER) of up to 10^{-3} , and turbo encoding yields a BER of up to 10^{-6} with computational complexity.

1.6.3 Digital Video Broadcasting Channel Coding

Convolutional codes concatenated with a Reed–Solomon (RS) code are adopted as physical layer FEC codes in digital video broadcast terrestrial/handheld (DVB-T/H). Turbo codes are used in DVB-SH (digital video broadcast satellite services to handhelds/terrestrial). Low-density parity check (LDPC) codes concatenated with a BCH (Bose–Chaudhuri–Hochquenghem) code are adopted as physical layer FEC in DVB-S2 (digital video broadcast second generation satellite) and DVB-T2 (digital video broadcast second generation terrestrial).

1.6.4 Error Correction Coding in GPS L5 Signal

A block diagram of the simplified GPS satellite L5 signal generator [8] is shown in Fig. 1.14. The navigation data is coded in a CRC block coder with a long block of 150 bits or 3 s at 50 bps and provides a 24-bit parity check word for low probability of undetected error. This bit stream is then rate $\frac{1}{2}$ coded using a $K = 7$ convolutional coder for error correction with a soft decision Viterbi decoder in the receiver. This FEC decoding provides approximately 5 dB of coding gain.

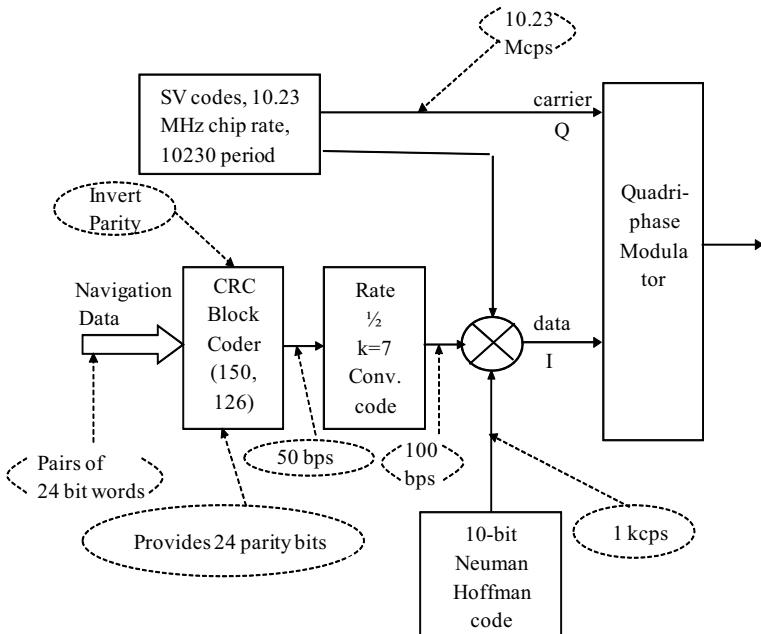


Fig. 1.14 Error correction coding in GPS L5 signal (copyright: 1999 ION)

References

1. E.N. Gilbert, Capacity of a burst noise channel. *Bell Syst. Techn. J.* **39**, 1253–1266 (1960)
2. E.O. Elliott, Estimates for error rates for codes on burst noise channels. *Bell Syst. Tech. J.* **42**, 1977–1997 (1963)
3. K.-L. Du, M.N.S. Swamy, *Wireless Communications: Communication Systems from RF Subsystems to 4G Enabling Technologies*. Cambridge University Press (2010)
4. M. Nakagami, The m-distribution: a general formula of intensity distribution of rapid fading, in *Statistical Methods in Radio Wave Propagation*, ed. by W.C. Hoffmann (Oxford Pergamon Press, 1960), pp. 3–36
5. V. Kuhn, *Wireless Communications Over MIMO Channels: Applications to CDMA and Multiple Antenna Systems* (Wiley, Chichester, UK, 2006)
6. E.W. Ryan, S. Lin, *Channel Codes Classical and Modern* (Cambridge University Press, New York, 2009)
7. A. Goldsmith, *Wireless Communications Cambridge* (Cambridge University Press, UK, 2005)
8. J.J. Spilker, A.J. Van Dierendonck, Proposed New Civil GPS Signal at 1176.45 MHz, in *ION GPS'99*, Nashville, TN, 4–17 September 1999

Chapter 2

Overview of the Performance of Digital Communication Over Fading Channels



In this chapter, bit error rate (BER) performance of some of digital modulation schemes and different wireless communication techniques is evaluated in AWGN and fading channels. Further, the BER performance of different diversity techniques such as selective diversity, EGC, and MRC is also evaluated in Rayleigh fading channel.

2.1 BER Performance of Different Modulation Schemes in AWGN, Rayleigh, and Rician Fading Channels

In this section, the effect of fading is evaluated on different modulation schemes. The bit error probability P_b which is often referred to as bit error rate (BER) is a better performance measure to evaluate a modulation scheme. The BER performance of any digital modulation scheme in a slow flat fading channel can be evaluated by the following integral:

$$P_b = \int_0^{\infty} P_{b,\text{AWGN}}(\gamma) P_{\text{df}}(\gamma) d\gamma \quad (2.1)$$

where $P_{b,\text{AWGN}}(\gamma)$ is the probability of error of a particular modulation scheme in AWGN channel at a specific signal-to-noise ratio

$$\gamma = h^2 \frac{E_b}{N_0},$$

where the random variable h is the channel gain, $\frac{E_b}{N_0}$ is the ratio of bit energy to noise power density in a non-fading AWGN channel, the random variable h^2 represents

the instantaneous power of the fading channel, and $P_{\text{df}}(\gamma)$ is the probability density function of γ due to the fading channel.

2.1.1 BER of BPSK Modulation in AWGN Channel

It is known that the bit error rate (BER) for M-PSK in additive white Gaussian noise (AWGN) channel is given by [1]

$$\text{BER}_{M-\text{PSK}} = \frac{2}{\max(\log_2 M, 2)} \sum_{k=1}^{\max(M/4, 1)} Q\left(\sqrt{\frac{2E_b \log_2 M}{N_0}} \sin \frac{(2k-1)\pi}{M}\right) \quad (2.2)$$

For coherent detection of BPSK, Eq. (2.2) with $M = 2$ reduces to

$$\text{BER}_{\text{BPSK}} = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \quad (2.3)$$

where $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty \exp\left(-\frac{y^2}{2}\right) dy$

Equation (2.3) can be rewritten as

$$\text{BER}_{\text{BPSK, AWGN}} = \frac{1}{2} \operatorname{erfc}\left(\sqrt{\frac{E_b}{N_0}}\right) \quad (2.4)$$

where erfc is the complementary error function and $\frac{E_b}{N_0}$ is the bit energy to noise ratio. The erfc can be related to the Q function as

$$Q(x) = \frac{1}{2} \operatorname{erfc}\left(\frac{x}{\sqrt{2}}\right) \quad (2.5)$$

Under the condition that the argument $\frac{E_b}{N_0} \geq 6.5\text{dB}$, Eq. (2.3) can be approximated as

$$\text{BER}_{\text{BPSK}} = \frac{\exp\left(-\frac{E_b}{N_0}\right)}{\sqrt{4\pi\left(\frac{E_b}{N_0}\right)}} \quad (2.5a)$$

For large $\frac{E_b}{N_0}$ and $M > 4$, the BER expression can be simplified as

$$\text{BER}_{\text{M-PSK}} = \frac{2}{\log_2 M} Q\left(\sqrt{\frac{2E_b \log_2 M}{N_0}} \sin \frac{\pi}{M}\right) \quad (2.6)$$

2.1.2 BER of BPSK Modulation in Rayleigh Fading Channel

For Rayleigh fading channels, h is Rayleigh distributed, and h^2 has chi-square distribution with two degrees of freedom. Hence,

$$P_{\text{df}}(\gamma) = \frac{1}{\bar{\gamma}} \exp\left(-\frac{\gamma}{\bar{\gamma}}\right) \quad (2.7)$$

where $\bar{\gamma} = \frac{E_b}{N_0} E[h^2]$ is the average signal-to-noise ratio. For $E[h^2] = 1$, $\bar{\gamma}$ corresponds to the average $\frac{E_b}{N_0}$ for the fading channel.

By using Eqs. (2.1) and (2.3), the BER for a slow Rayleigh fading channel with BPSK modulation can be expressed as [2, 3]

$$\text{BER}_{\text{BPSK,Rayleigh}} = \frac{1}{2} \left(1 - \sqrt{\frac{\bar{\gamma}}{1 + \bar{\gamma}}} \right) \quad (2.8)$$

For $E[h^2] = 1$, Eq. (2.8) can be rewritten as

$$\text{BER}_{\text{BPSK,Rayleigh}} = \frac{1}{2} \left(1 - \sqrt{\frac{\frac{E_b}{N_0}}{1 + \frac{E_b}{N_0}}} \right) \quad (2.9)$$

For high $\frac{E_b}{N_0}$, Eq. (2.9) can be approximated as

$$\begin{aligned} \text{BER}_{\text{BPSK,Rayleigh}} &= \frac{1}{2} \left(1 - \sqrt{\frac{\frac{E_b}{N_0}}{1 + \frac{E_b}{N_0}}} \right) = \frac{1}{2} \left(1 - \frac{1}{\sqrt{1 + \frac{2}{\frac{E_b}{N_0}}}} \right) \\ &\approx \frac{1}{2} \left[1 - \left(1 - \frac{1}{2} \left(\frac{2}{\frac{E_b}{N_0}} \right) \right) \right] = \frac{1}{\frac{2E_b}{N_0}} \end{aligned} \quad (2.9a)$$

2.1.3 BER of BPSK Modulation in Rician Fading Channel

The error probability estimates for linear BPSK signaling in Rician fading channels are well documented in [4] and are given as

$$P_{b,\text{Rician}} = Q_1(a, b) - \frac{1}{2} \left[1 + \sqrt{\frac{d}{d+1}} \right] \exp\left(-\frac{a^2 + b^2}{2}\right) I_0(ab) \quad (2.10)$$

where

$$a = \left[\sqrt{\frac{K_r^2[1 + 2d - 2\sqrt{d(d+1)}]}{2(d+1)}} \right], b = \left[\sqrt{\frac{K_r^2[1 + 2d + 2\sqrt{d(d+1)}]}{2(d+1)}} \right]$$

$$K_r = \frac{\sigma^2}{2\sigma^2}, d = \sigma^2 \frac{E_b}{N_0}.$$

The parameter K_r is the Rician factor.

The $Q_1(a, b)$ is the Marcum Q function defined [2] as

$$\begin{aligned} Q_1(a, b) &= \exp\left(-\frac{a^2 + b^2}{2}\right) \sum_{l=0}^{\infty} \left(\frac{a}{b}\right)^l I_0(ab), \quad b \geq a > 0 \\ Q_1(a, b) &= Q(b-a), \quad b \gg 1 \text{ and } b \gg b-a \end{aligned} \quad (2.11)$$

The following MATLAB program is used to illustrate the BER performance of BPSK in AWGN, Rayleigh, and Rician fading channels.

Program 2.1: Program for computing the BER for BPSK modulation in AWGN, Rayleigh, and Rician fading channels

```
clear all;
clc;
M=2;K=5;DIVORDER= 1;
EbNo = 0:5:35;
BER_Ray = BERFADING(EbNo, 'psk', M, DIVORDER);
BER_Rician = BERFADING(EbNo, 'psk', 2, 1, K);
BER = BERAWGN(EbNo, 'psk', M,'nondiff');
semilogy(EbNo,BER,'o-');
hold on
semilogy(EbNo,BER_Ray,'*-');
semilogy(EbNo,BER_Rician,'+-');grid on
legend('AWGN channel','Rayleighchannel','Rician channel');%, 'Rayleigh-Simulation';
xlabel('Eb/No, dB');
ylabel('Bit Error Rate');
axis([ 0 35 1e-5 1 ])
```

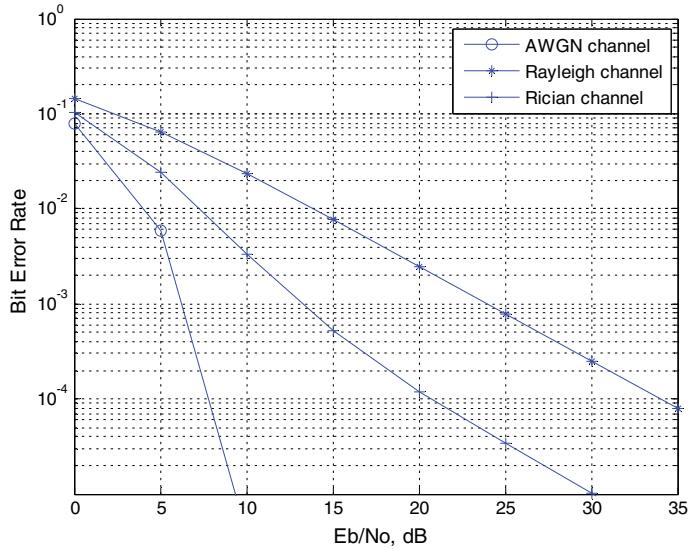


Fig. 2.1 BER performance of BPSK in AWGN, Rayleigh, and Rician fading channels

The BER performance resulted from the above MATLAB program for BPSK in the AWGN, Rayleigh, and Rician ($K_r = 5$) channels is depicted in Fig. 2.1.

From Fig. 2.1, for instance, we can see that to obtain a BER of 10^{-4} , using BPSK, an AWGN channel requires $\frac{E_b}{N_0}$ of 8.35 dB, Rician channel requires $\frac{E_b}{N_0}$ of 20.5 dB, and a Rayleigh channel requires $\frac{E_b}{N_0}$ of 34 dB. It is clearly indicative of the large performance difference between AWGN channel and fading channels.

2.1.4 BER Performance of BFSK in AWGN, Rayleigh, and Rician Fading Channels

In BPSK, the receiver provides coherent phase reference to demodulate the received signal, whereas the certain applications use non-coherent formats avoiding a phase reference. This type of non-coherent format is known as non-coherent binary phase-shift keying (BFSK).

The BER for non-coherent BFSK in slow flat fading Rician channel is expressed as [3]

$$P_{b,\text{BFSK(Ric)}} = \frac{1 + K_r}{2 + 2K_r + \bar{\gamma}} \exp\left(-\frac{K_r \bar{\gamma}}{2 + 2K_r + \bar{\gamma}}\right) \quad (2.12)$$

where K_r is the power ratio between the LOS path and non-LOS paths in the Rician fading channel.

Substituting $K_r = \infty$ in Eq. (2.12), the BER in AWGN channel for non-coherent BFSK can be expressed as

$$P_{b,\text{AWGN}} = \frac{1}{2} \exp\left(-\frac{E_b}{2N_0}\right) \quad (2.13)$$

whereas substitution of $K_r = 0$ leads to the following BER expression for slow flat Rayleigh fading channels using non-coherent BFSK modulation

$$P_{b,\text{BFSK(Ray)}} = \frac{1}{2 + \bar{\gamma}} \quad (2.14)$$

The following MATLAB program is used to illustrate the BER performance of non-coherent BFSK modulation in AWGN, Rayleigh, and Rician fading channels.

Program 2.2: Program for computing the BER for BFSK modulation in AWGN, Rayleigh, and Rician fading channels

```
clear all;
clc;
Eb_N0_dB = [0:5:35];
K=5;
EbN0Lin = 10.^{(Eb_N0_dB/10)};
theoryBerAWGN = 0.5*exp(-0.5*EbN0Lin); % theoretical ber
for i=1:8
    theoryBer(i) = 1/(EbN0Lin(i)+2);
    theoryberric(i)=((1+K)/(EbN0Lin(i)+2+2*K))*exp(-
    K*EbN0Lin(i)/(EbN0Lin(i)+2+2*K));
end
semilogy(Eb_N0_dB, theoryBerAWGN, '-o', 'LineWidth', 2);
hold on
semilogy(Eb_N0_dB, theoryBer, '-*', 'LineWidth', 2);
semilogy(Eb_N0_dB, theoryberric, '-+', 'LineWidth', 2);
axis([0 35 10^-6 0.5])
grid on
legend('AWGN channel', 'Rayleigh channel', 'Rician channel');%, 'Rayleigh-
Simulation');
xlabel('Eb/No, dB');
ylabel('Bit Error Rate');
```

The BER performance resulted from the MATLAB program 2.2 for non-coherent BFSK in the AWGN, Rayleigh, and Rician ($K_r = 5$) channels is depicted in Fig. 2.2.

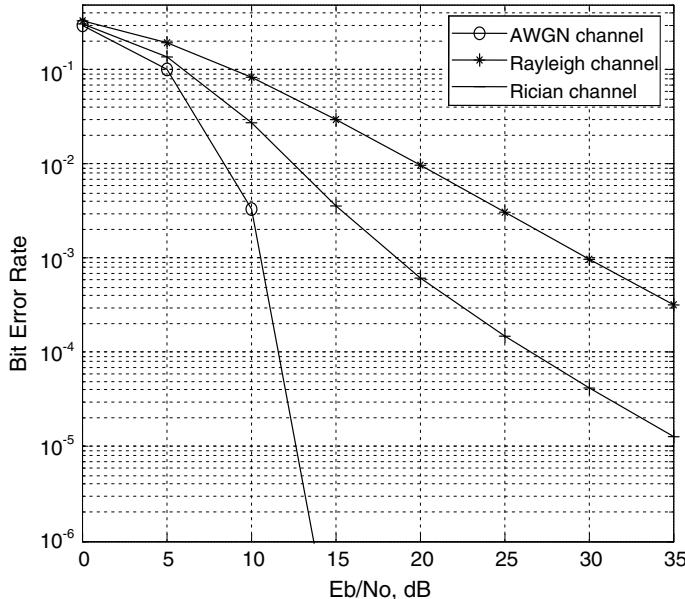


Fig. 2.2 BER performance of BFSK in AWGN, Rayleigh, and Rician fading channels

2.1.5 Comparison of BER Performance of BPSK, QPSK, and 16-QAM in AWGN and Rayleigh Fading Channels

The BER of gray-coded M-QAM in AWGN channel can be more accurately computed by [5]

$$\text{BER}_{16\text{-QAM, AWGN}} \approx \frac{4}{\log_2 M} \left(1 - \frac{1}{\sqrt{M}}\right) \sum_{i=1}^{\frac{\sqrt{M}}{2}} (2i-1) Q\left(\sqrt{\frac{3 \log_2 M E_b}{(M-1) N_0}}\right) \quad (2.15)$$

In Rayleigh fading, the average bit error rate for M-QAM is given by [6]

$$\text{BER}_{\text{MQAM, AWGN}} \approx \frac{2}{\log_2 M} \left(1 - \frac{1}{\sqrt{M}}\right) \sum_{i=1}^{\frac{\sqrt{M}}{2}} \left(1 - \sqrt{\frac{1.5(2i-1)^2 \bar{\gamma} \log_2 M}{M-1 + 1.5(2i-1)^2 \bar{\gamma} \log_2 M}}\right) \quad (2.16)$$

The following MATLAB program 2.3 is used to compute theoretic BER performance of 4-QAM, 8-QAM, and 16-QAM modulations in AWGN and Rayleigh fading channels.

Program 2.3: Program for computing theoretic BER for 4-QAM, 8-QAM, and 16-QAM modulations in AWGN and Rayleigh fading channels

```

clear all;
Eb_N0_dB = [0:35]; % multiple Eb/N0 values
EbN0Lin = 10.^{(Eb_N0_dB/10)};
M=4;
BER_4QAM_AWGN = 0.5* erfc( sqrt( EbN0Lin ) );
BER_4QAM_Rayleigh = 0.5.*(1-1*(1+1./EbN0Lin).^( -0.5));
M=8;
BER_8QAM_AWGN = 4/log2(M) * (1-1/sqrt(M))*( 0.5*erfc (sqrt( 3/2*
log2(M) *EbN0Lin / (M-1) )) + ... 0.5 * erfc (3* sqrt( 3/2* log2(M)
*EbN0Lin / (M-1) )) );
BER_8QAM_Rayleigh = (2/log2(M)) * (1 - 1/sqrt(M))*((1-
1*(1+7./(4.5*EbN0Lin)).^( -0.5))+...(1-1*(1+7./(40.5*EbN0Lin)).^( -0.5)));
M=16;
BER_16QAM_AWGN = 4/log2(M) * (1 - 1/sqrt(M)) * ( 0.5 * erfc ( sqrt(
3/2* log2(M) *EbN0Lin / (M-1) )) + ...0.5 * erfc (3* sqrt( 3/2* log2(M)
*EbN0Lin / (M-1) )) );
BER_16QAM_Rayleigh = 2/log2(M) * (1 - 1/sqrt(M))*((1-
1*(1+15./(6*EbN0Lin)).^( -0.5))+...(1-1*(1+15./(54*EbN0Lin)).^( -0.5)));
close all
Figure
semilogy(Eb_N0_dB,BER_16QAM_Rayleigh,'*', 'LineWidth',2);
hold on
semilogy(Eb_N0_dB,BER_8QAM_Rayleigh ,'-', 'LineWidth',2);
semilogy(Eb_N0_dB,BER_4QAM_Rayleigh ,'-x', 'LineWidth',2);
semilogy(Eb_N0_dB,BER_16QAM_AWGN,'--', 'LineWidth',2);
semilogy(Eb_N0_dB,BER_8QAM_AWGN,'-v', 'LineWidth',2);
semilogy(Eb_N0_dB,BER_4QAM_AWGN ,'-d', 'LineWidth',2);
axis([0 35 10^-8 1])
grid on
legend('16-QAM Rayleigh','8-QAM Rayleigh','4-QAM Rayleigh','16QAM
AWGN','8QAM AWGN', '4-QAM AWGN');
xlabel('Eb/No, dB');
ylabel('BER');

```

The BER performance obtained from the above program is depicted in Fig. 2.3.

2.2 Wireless Communication Techniques

The most known wireless communication techniques are:

Direct-sequence code division multiple access (DS-CDMA)

Frequency hopping CDMA (FH-CDMA)

Orthogonal frequency division multiplexing (OFDM)

Multicarrier CDMA (MC-CDMA).

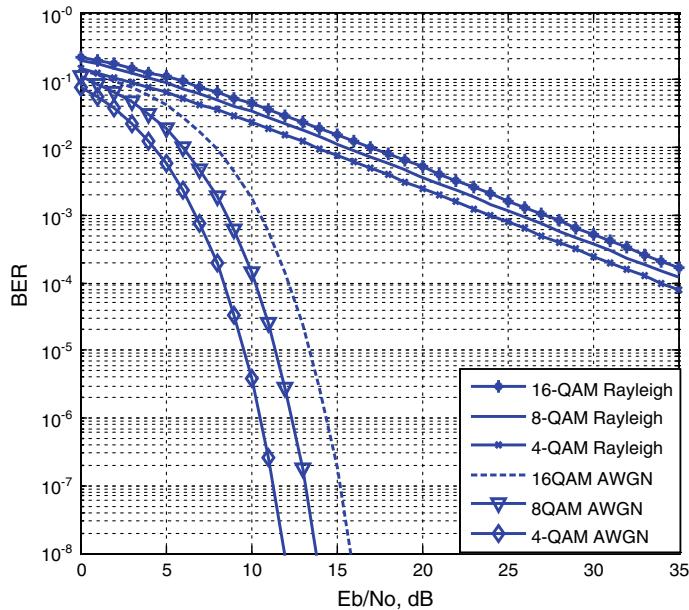


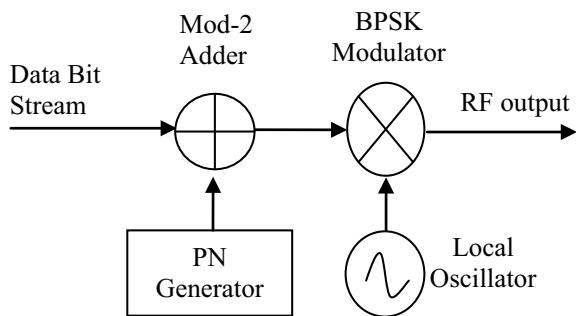
Fig. 2.3 BER performances of 4-QAM, 8-QAM, and 16-QAM in AWGN and Rayleigh fading channels

2.2.1 DS-CDMA

In code division multiple access (CDMA) systems, the narrowband message signal is multiplied by a very high bandwidth signal, which has a high chip rate; i.e., it accommodates more number of bits in a single bit of message signal. The signal with a high chip rate is called as spreading signal. All users in the CDMA system use the same carrier frequency and transmit simultaneously. The spreading signal or pseudo-noise code must be random so that no other user could be recognized. The intended receiver works with same PN code which is used by the corresponding transmitter, time correlation operation detects the specific desired code word only, and all other code words appear as noise. Each user operates independently with no knowledge of the other users.

The near-far problem occurs due to the sharing of the same channel by many mobile users. At the base station, the demodulator is captured by the strongest received mobile signal raising the noise floor for the weaker signals and decreasing the probability of weak signal reception. In most of the CDMA applications, power control is used to combat the near-far problem. In a cellular system, each base station provides power control to assure same signal level to the base station receiver from each mobile within the coverage area of the base station and solves the overpowering to the base station receiver by a nearby user drowning out the signals of faraway users.

Fig. 2.4 Scrambler system using BPSK modulation



In CDMA, the actual data is mixed with the output of a PN coder to perform the scrambling process. The scrambled data obtained after scrambling process is then modulated using BPSK or QPSK modulator as shown in Fig. 2.4. The BPSK or QPSK modulated data is then transmitted.

2.2.1.1 BER Performance of DS-CDMA in AWGN and Rayleigh Fading Channels

Let us consider a single cell with K users with each user having a PN sequence length N chips per message symbol. The received signal will consist of the sum of the desired user signal, $K - 1$ undesired user transmitted signals, and additive noise. Approximating the total multiple access interference caused by the $K - 1$ users as a Gaussian random variable, the BER for DS-CDMA in AWGN channel is given [3] by

$$P_{b,\text{CDMA(AWGN)}} = Q\left(\frac{1}{\sqrt{\frac{K-1}{3N} + \frac{N_0}{2E_b}}}\right) \quad (2.17)$$

The BER for DS-CDMA in Rayleigh fading channel can be expressed [7] as

$$P_{b,\text{CDMA(Ray)}} = \frac{1}{2}\left(1 - \frac{1}{\sqrt{1 + \frac{N_0}{2E_b\sigma^2} + \frac{K-1}{3N}}}\right) \quad (2.18)$$

where σ^2 is the variance of the Rayleigh fading random variable.

The following MATLAB program is used to compute theoretic BER of DS-CDMA in AWGN and Rayleigh fading channels.

Program 2.4: Program to compute BER performance of DS-CDMA in AWGN and Rayleigh fading channels

```

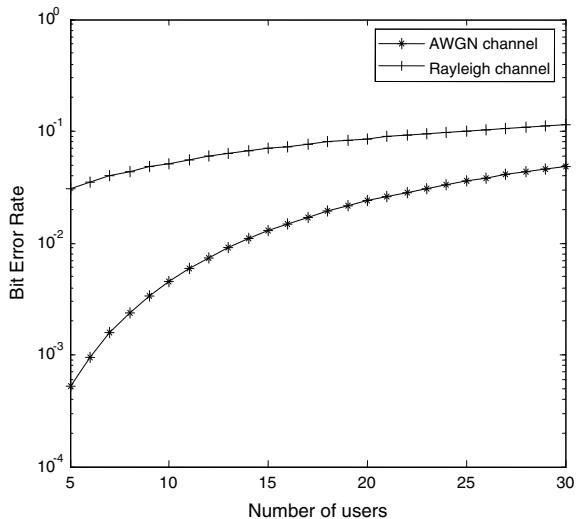
clearall;clc;close all;
Eb_N0_dB=10;
EbN0Lin = 10.^{(Eb_N0_dB/10)};
N=31;
for k=3:30
xx(k)=1/sqrt(((k-1)/(3*N))+0.5*1/(EbN0Lin));
xxf(k)=sqrt(1+((k-1)/N)+0.5*1/(EbN0Lin));
bercdma(k)=0.5*erfc(xx(k)/sqrt(2));
bercdmaf(k)=0.5-0.5/xxf(k);
end
semilogy(5:30,bercdma(5:30),'-*')
hold on
semilogy(5:30,bercdmaf(5:30),'-+')
legend('AWGN channel','Rayleigh channel');
xlabel('Number of users');
ylabel('Bit Error Rate');

```

The BER performance from the above program for DS-CDMA in the AWGN and Rayleigh channels for $N = 31$, $\sigma^2 = 1$, and $\frac{E_b}{N_0} = 20$ dB is depicted in Fig. 2.5.

From Fig. 2.5, it is observed that the BER performance of DS-CDMA is better in AWGN channel as compared to Rayleigh fading channel. Further, with an increased number of users, the BER performance decreases in both the channels.

Fig. 2.5 BER performance of DS-CDMA in AWGN and Rayleigh fading channels for $N = 31$, $\sigma^2 = 1$ and $\frac{E_b}{N_0} = 10$ dB



2.2.2 FH-CDMA

In FH-CDMA, each data bit is divided over a number of frequency-hop channels (carrier frequencies). At each frequency-hop channel, a complete PN sequence of length N is combined with the data signal. Applying fast frequency hopping (FFH) requires a wider bandwidth than slow frequency hopping (SFH). The difference between the traditional slow and fast frequency hopping schemes can be visualized as shown in Fig. 2.6. A slow hopped system has one or more information symbols per hop or slot. It is suitable for high-capacity wireless communications. A fast hopped system has the hopping rate greater than the data rate. During one information symbol, the system transmits over many bands with short duration. It is more prevalent in military communications.

In FH-CDMA, modulation by some kind of the phase-shift keying is quite susceptible to channel distortions due to several frequency hops in each data bit. Hence, an FSK modulation scheme is to be chosen for FH-CDMA.

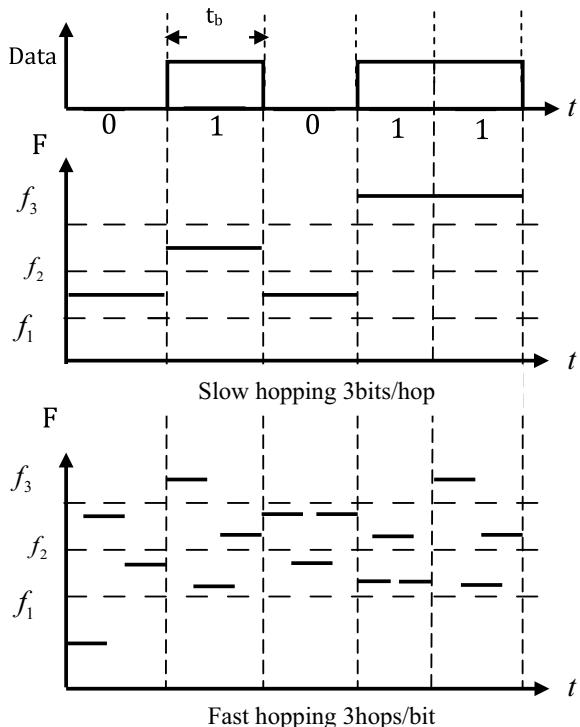
The hop set, dwell time, and hop rate with respect to FH-CDMA are defined as

Hop set: It is the number of different frequencies used by the system.

Dwell time: It is defined as the length of time that the system spent on one frequency for transmission.

Hop rate: It is the rate at which the system changes from one frequency to another.

Fig. 2.6 Slow and fast hopping



2.2.2.1 BER Expression for Synchronous SFH-CDMA

Consider a SFH-CDMA channel with K active users and q (frequency) slots. The hit probability is the probability that a number of interfering users are transmitting on the same frequency-hop channel as the reference user. This probability will be referred to as $P_h(K)$, where K is the total number of active users.

The probability of hitting from a given user is given by [8]

$$P = \frac{1}{q} \left(1 + \frac{1}{N_b} \right) \quad (2.19)$$

where N_b is the number of bits per hop, and q stands for the number of hops. The primary interest for our analysis is the probability P_h of one or more hits from the $K - 1$ users which is given by

$$P_h = 1 - (1 - P)^{K-1} \quad (2.20)$$

By substituting “ P ” value from Eq. (2.19) in Eq. (2.20), we get the probability of hit from $K - 1$ users as

$$P_h(K) = 1 - \left(1 - \frac{1}{q} \left(1 + \frac{1}{N_b} \right) \right)^{K-1} \quad (2.21)$$

If it is assumed that all users hop their carrier frequencies synchronously, the probability of hits is given by

$$P_h = 1 - \left(1 - \frac{1}{q} \right)^{K-1} \quad (2.22)$$

For large q ,

$$P_h(K) = 1 - \left(1 - \frac{1}{q} \right)^{K-1} \approx \frac{K-1}{q} \quad (2.23)$$

The probability of bit error for synchronous MFSK SFH-CDMA when the K number of active users is present in the system can be found by [9]

$$P_{\text{SFH}}(K) = \sum_{k=1}^K \binom{K-1}{k} P_h^k (1 - P_h)^{K-1-k} P_{\text{MFSK}}(K) \quad (2.24)$$

where $P_{\text{MFSK}}(K)$ denotes the probability of error when the reference user is hit by all other active users. Equation (2.24) is the upper bound of the bit error probability of the SFH-CDMA system. The $P_{\text{MFSK}}(K)$ for the AWGN and flat fading channels can be expressed as [10]

$$P_{\text{MFSK}}(K) = \begin{cases} \sum_{i=1}^{M-1} \frac{(-1)^{i+1}}{i+1} \binom{M-1}{i} \exp\left(-\frac{i \frac{E_b}{N_0}}{i+1}\right) & \text{AWGN} \\ \sum_{i=1}^{M-1} \frac{(-1)^{i+1}}{1+i+i \frac{E_b}{N_0}} \binom{M-1}{i} & \text{Rayleigh fading} \end{cases} \quad (2.25)$$

The following MATLAB program computes theoretic BER of SFH-CDMA in AWGN and Rayleigh fading channels.

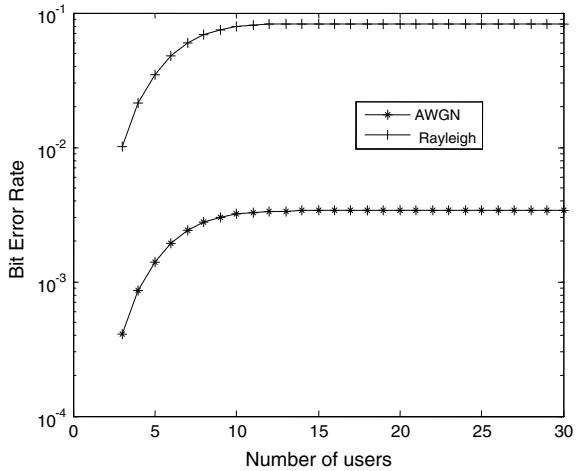
Program 2.5: Program to compute BER performance of SFH-CDMA in AWGN and Rayleigh fading channels

```

clearall;clc;
snr1=10;Eb_N0_dB=10;
EbNOLin = 10.^^(Eb_N0_dB/10);
q=32;
pe=0.5*exp(-(EbNOLin /2));
for K=3:30
    ph=(K-1)/q;
    pe1=0;
    for k=1:(K-1)
        pe1=pe1+nchoosek(K-1,k)*(ph)^k*(1-ph)^(K-1-k);
    end
    pesfh(K)=pe1*pe;
    end
    disp(pesfh);
    semilogy(3:30,pesfh(3:30),'-*');
    hold on;
    pe=1/(2+EbNOLin );
    for K=3:30
        ph=(K-1)/q;
        pe1=0;
        for k=1:(K-1)
            pe1=pe1+nchoosek(K-1,k)*(ph)^k*(1-ph)^(K-1-k);
        end
        pesfhr(K)=pe1*pe;
        end
        disp(pesfhr);
        semilogy(3:30,pesfhr(3:30),'-+');
        hold on;
        legend('AWGN' , ' Rayleigh');
        xlabel('Number of users');
        ylabel('Bit Error Rate');

```

Fig. 2.7 BER performance of SFH-CDMA in AWGN and Rayleigh fading channels with $q = 32$, $M = 2$ (BFSK) at $\frac{E_b}{N_0} = 10$ dB



The BER performance from the above program for SFH-CDMA in the AWGN and Rayleigh channels with $q = 32$, $M = 2$ (BFSK) at $\frac{E_b}{N_0} = 10$ dB is depicted in Fig. 2.7.

2.2.3 OFDM

The block diagram of OFDM transmitter is shown in Fig. 2.8. In OFDM, the input data is serial-to-parallel converted (the S/P block). Then, the inverse fast Fourier transform (IFFT) is performed on the N parallel outputs of the S/P block to create an OFDM symbol. The complex numbers in the output of the IFFT block are parallel-to-serial converted (P/S). Then, the cyclic prefix is inserted in order to combat the intersymbol interference (ISI) and intercarrier interference (ICI) caused by the multipath channel. To create the cyclic prefix, the complex vector of length at the end of the symbol duration T is copied and appended to the front of the signal block as shown in Fig. 2.9.

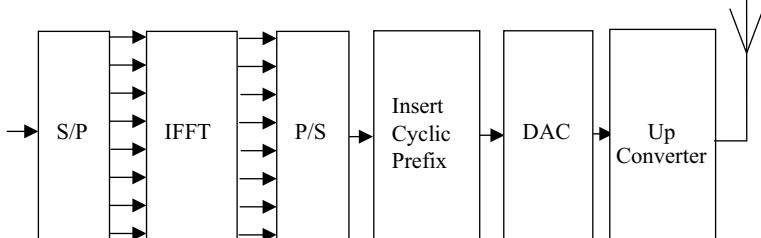


Fig. 2.8 Schematic block diagram of OFDM transmitter

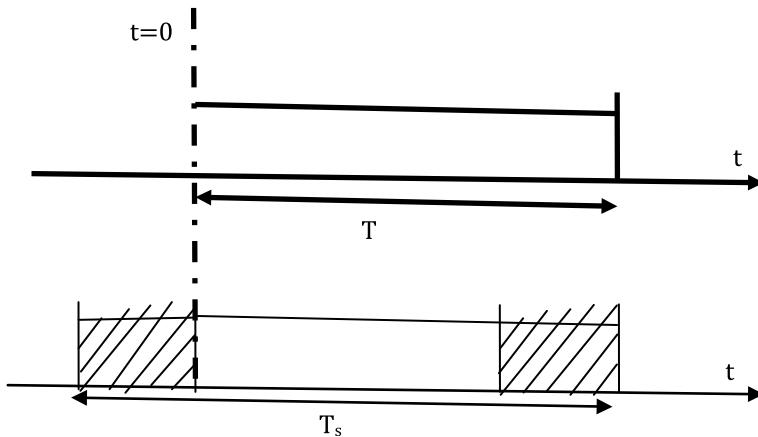


Fig. 2.9 Inserting cyclic prefix

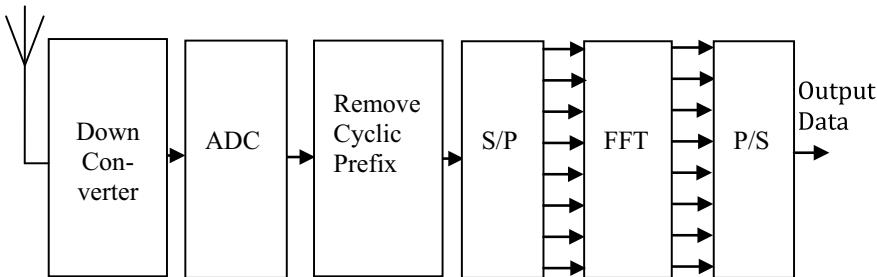


Fig. 2.10 Schematic block diagram of OFDM receiver

The schematic block diagram of the OFDM receiver is shown in Fig. 2.10. It is the exact inverse of the transmitter shown in Fig. 2.8.

In wireless communication, channel estimation is a vital requirement for channel equalization and signal detection. In various wireless propagation environments, the channel may consist of only a few dominant propagation (nonzero) paths, even though it has a large delay spread. Thus, the channel impulse response has a sparse nature. However, conventional methods, such as least squares (LS), ignore this prior information about the unknown channel leading to lower spectral efficiency. Orthogonal matching pursuit (OMP) algorithm is one of the most known sparse signal recovery algorithms.

The following MATLAB program 2.6 is used to compute BER performance (theoretic, with the known channel, with estimated channel (OMP) algorithm) OFDM in sparse Rayleigh fading channel using 4-QAM modulation.

Program 2.6: Program to compute BER performance of OFDM in sparse Rayleigh fading channel

```

close all; clear all;
N_fft=512;% FFT length
N_g=64 N_ofdm=N_fft+N_g; N_sym=100;
N_ps=8; N_p=N_fft/N_ps; N_d=N_fft-N_p; % Pilot spacing, Numbers
of pilots and data per OFDM symbol
N_bps=2;M=2^N_bps; % Number of bits per (modulated) symbol
mod_object = modem.qammod('M',M, 'SymbolOrder','gray');
demod_object = modem.qamdemod('M',M, 'SymbolOrder','gray');
Es=1; A=sqrt(3/2/(M-1)*Es); % Signal energy& QAM normalization
factor
EbN0s = [0:5:25];
rand('seed',1)
randn('seed',1)
L = 64; %Total number of channel taps
% -- Sparse signal "h11"
K =7;% 16; % non-zero channel taps,SPARSITY
T = randperm(L);
T = T(1:K);
h= zeros(L,1);
h(T) = randn(K,1) + li*randn(K,1);
H= fft(h',N_fft); channel_length = length(h); % True channel and
its time-domain length
for i=1:length(EbN0s)
EbN0 = EbN0s(i);
rand('seed',1)
randn('seed',1);
BER_ZFi=0;
BER_OMP_ZFi=0;
for nsym=1:N_sym
X_p= 2*(randn(1,N_p)>0)-1; % Pilot sequence generation
msg_int=randi(1,N_fft-N_p,M); % bit generatio
Data = modulate(mod_object,msg_int)*A; %QAM Modulated symbols
ip = 0; pilot_loc = [];
for k=1:N_fft
if mod(k,N_ps)==1
X(k) = X_p(floor(k/N_ps)+1);
pilot_loc = [pilot_loc k];
ip = ip+1;
else
X(k) = Data(k-ip);
end
end
x = ifft(X,N_fft); % IFFT
x = [x(N_fft-N_g+1:N_fft) x]; % Add CP
pilot_loc = setxor(1:N_fft,pilot_loc);% Data locations in the subcarriers
seed = 0;
y = conv(x, h'); % Channel path (convolution)
y= awgn(y,EbN0,'measured'); %awgn noise added
y = y(N_g+1:N_ofdm); % Remove CP
Y = fft(y);% FFT
yps=Y(pilot_loc);
for ii=1:N_p
ypss(ii,1)=yps(ii);
end

```

```

W =exp(-j*2*pi/N_fft*[0:N_fft-1]'*[0:N_fft-1]);
WW=W(1:N_ps:N_fft,1:L); %sub dft matrix
AA=diag(X_p)*WW;
homp=ompgrs(ypss,AA,K);
H_Sparse=fft(homp,N_fft);
for i1=1:N_fft
    H_Sparse(1,i1)=H_Sparse(i1);
end
Y_ZF_OMP = (Y(ofdm.DP1)./H_Sparse(ofdm.DP1));
Y_ZF = (Y(ofdm.DP1)./H(ofdm.DP1));
Data_extracted_ZF= Y_ZF;% data extracted with known channel
Data_extracted_OMP_ZF= Y_ZF_OMP;% data extracted with chan-
nelestimation using LS
msg_detected_ZF=qamdemod(Data_extracted_ZF/A,M);% message in-
formation bits detected with known channel
msg_detected_OMP_ZF=qamdemod(Data_extracted_OMP_ZF/A,M);% mes-
sage information bits detected with estimatedchannel
BER_ZFi = BER_ZFi +
biterr(abs(round(msg_detected_ZF)),msg_int');%BER with known
channel
BER_OMP_ZFi = BER_OMP_ZFi +
biterr(abs(round(msg_detected_OMP_ZF)),msg_int');% BER with
estimated channel
end
BER_ZF(i)= mean(BER_ZFi);
BER_OMP_ZF(i)= mean(BER_OMP_ZFi);
hold on
semilogy( EbN0s',BER_ZF,'-o');
semilogy( EbN0s',BER_OMP_ZF,'-d');
semilogy( EbN0s',BER_4QAM_Rayleigh_theretic ,'-s');
legend('with known channel','with estimated channel using
OMP','theoretic')
xlabel ('Eb/N0 (dB)') ;
ylabel ( 'BER')

```

MATLAB function for OMP algorithm

```

function homp=ompqgs(ypss,AA,K)
rk=ypss;A=AA;
[a,b]=size(A);
estSupportSet = [];
for i=1:K
    matchFilter = abs(A'*rk);
    [~, index] = max(matchFilter);
    estSupportSet = union(estSupportSet, index);
    A1=A(:,estSupportSet);
    [m,n]=size(A1);
    Q=zeros(m,n);R=zeros(n,n);
    for jj=1:n
        v=A1(:,jj);
        for i=1:jj-1
            R(i,jj)=Q(:,i)'*v;
            v=v-R(i,jj)*Q(:,i);
        end
        R(jj,jj)=norm(v);
        Q(:,jj)=v/R(jj,jj);
    end
    rk=rk-Q'*Q'*rk;
end
bb=Q'*ypss;n = length( bb );
xhat = zeros( n, 1 );
for i=n:-1:1
    xhat(i) = ( bb(i) - R(i, :) * xhat ) / R(i, i);
end
homp=zeros(1,b);
for i=1:K
    homp(estSupportSet(i))=xhat(i);
end

```

The BER performance from the above program of OFDM in sparse Rayleigh fading channel using 4-QAM is shown in Fig. 2.11.

2.2.4 MC-CDMA

MC-CDMA is a combination of OFDM and CDMA having the benefits of both OFDM and CDMA. In MC-CDMA, frequency diversity is achieved by modulating symbols on many subcarriers instead of modulating on one carrier like in CDMA.

In MC-CDMA, the same symbol is transmitted through many subcarriers in parallel, whereas in OFDM, different symbols are transmitted on different subcarriers. The block diagram of the MC-CDMA system transmitter is shown in Fig. 2.12. The block diagram of the MC-CDMA system receiver is shown in Fig. 2.13. In the receiver, the cyclic prefix is removed and FFT is performed to obtain the signals in the frequency domain.

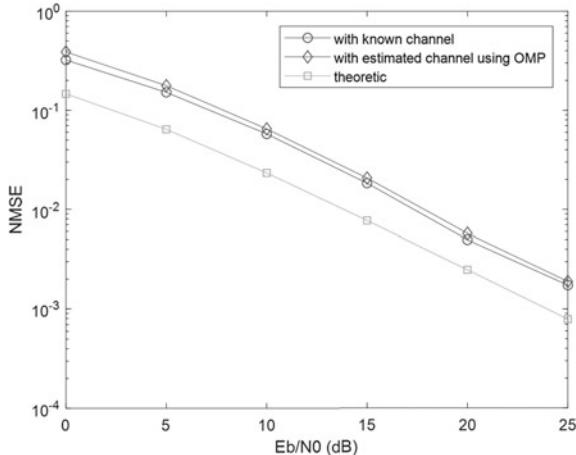


Fig. 2.11 BER performance of OFDM in sparse Rayleigh fading channel

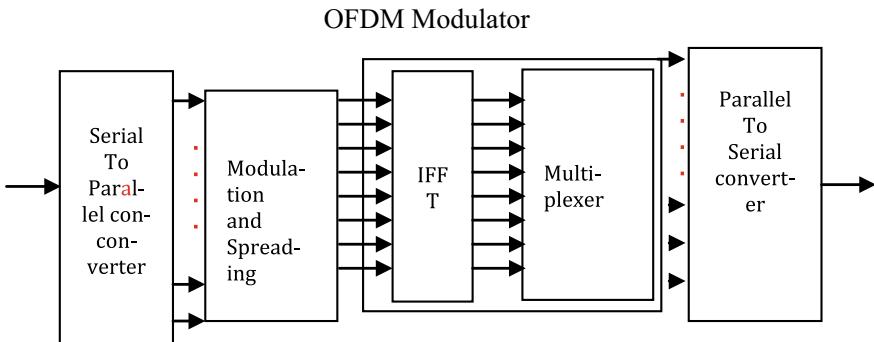


Fig. 2.12 Block diagram of MC-CDMA transmitter

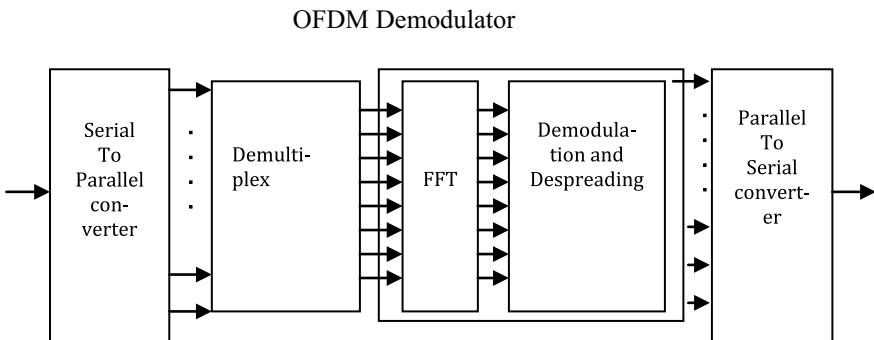


Fig. 2.13 Block diagram of MC-CDMA receiver

2.2.4.1 BER Expression for Synchronous MC-CDMA

Assuming a synchronous MC-CDMA system with k users, N subcarriers, binary phase-shift keying (BPSK) modulation, the BER for MC-CDMA in slowly varying Rayleigh fading channel can be calculated using the residue method by [11]

$$P_{\text{MC-CDMA, Rayleigh}}(k) = \frac{(2c)^{N_c}}{[(N_c-1)!]^2} \sum_{k=0}^{N_c-1} \binom{N_c-1}{k} (N_c-1-k)! (N_c-1-k)!(c+d)^{-(N_c-k)} (2d)^{-(N_c+k)} \quad (2.26)$$

where k stands for the number of users, N_c denotes the number of subcarriers, and the parameters c and d are defined by

$$\frac{1}{2c} = \frac{N_c}{4E_b/N_0} + \frac{k+1}{4}, d = \sqrt{c^2 + 2c} \quad (2.27)$$

A theoretical BER performance comparison of DS-CDMA, SFH-CDMA, and MC-CDMA in Rayleigh fading channels at $\frac{E_b}{N_0} = 10$ dB is shown in Fig. 2.14.

From Fig. 2.14, it is observed that MC-CDM outperforms both the DS-CDMA and SFH-CDMA.

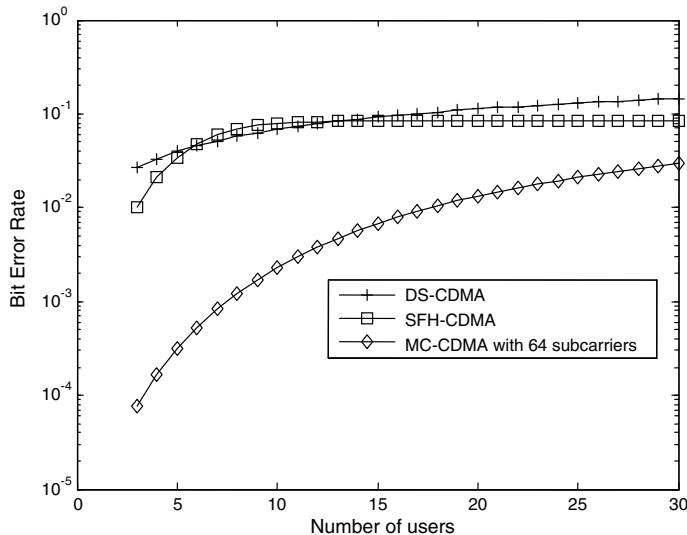


Fig. 2.14 BER performance of DS-CDMA, SFH-CDMA, and MC-CDMA in Rayleigh fading channels at $\frac{E_b}{N_0} = 10$ dB

2.3 Diversity Reception

Two channels with different frequencies, polarizations, or physical locations experience fading independently of each other. By combining two or more such channels, fading can be reduced. This is called diversity.

On a fading channel, the SNR at the receiver is a random variable; the idea is to transmit the same signal through r separate fading channels. These are chosen so as to provide the receiver with r independent (or close-to-independent) replicas of the same signal, giving rise to independent SNRs. If r is large enough, then, at any time instant, there is a high probability that at least one of the signals received from the r “diversity branches” is not affected by a deep fade and hence that its SNR is above a critical threshold. By suitably combining the received signals, the fading effect will be mitigated (Fig. 2.15).

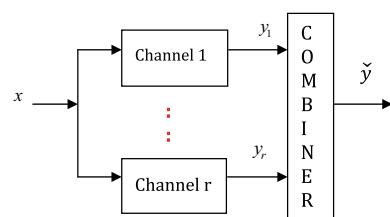
Many techniques have been advocated for generating the independent channels on which the diversity principle is based, and several methods are known for combining the signals y_1, \dots, y_r obtained at their outputs into a single channel \tilde{y} . Among the categorized techniques, the most important ones are

- (1) Space diversity
- (2) Polarization diversity
- (3) Frequency diversity
- (4) Time diversity
- (5) Cooperative diversity.

Space diversity: To obtain sufficient correlation, the spacing between the r separate antennas should be wide with respect to their coherent distance while receiving the signal. It does not require any extra spectrum occupancy and can be easily implemented.

Polarization diversity: Over a wireless channel, multipath components polarized either horizontally or vertically have different propagation. Diversity is provided when the receiving signal uses two different polarized antennas. In another way, two cross-polarized antennas with no spacing between them also provide diversity. Cross-polarized antennas are preferred since they are able to double the antenna numbers using half the spacing being used for co-polarized antennas. Polarized diversity can achieve more gain than space diversity alone in reasonable scattering areas, and hence, it is deployed in more and more BSs.

Fig. 2.15 Diversity and combining



Frequency diversity: In order to obtain frequency diversity, the same signal over different carrier frequencies should be sent whose separation must be larger than the coherence bandwidth of the channel.

Time diversity: This is obtained by transmitting the same signal in different time slots separated by a longer interval than the coherence time of the channel.

Cooperative diversity: This is obtained by sharing of resources by users or nodes in a wireless network and transmits cooperatively. The users or nodes act like an antenna array and provide diversity. This type of diversity can be achieved by combining the signals transmitted from the direct and relay links.

2.3.1 Receive Diversity with N Receive Antennas in AWGN

The received signal on the i th antenna can be expressed as

$$y_i = h_i x + \eta_i \quad (2.28)$$

where

y_i is the symbol received on the i th receive antenna,

h_i is the channel gain on the i th receive antenna,

x is the input symbol transmitted, and

η_i is the noise on the i th receive antenna.

The received signal can be written in matrix form as

$$y = hx + n$$

where

$y = [y_1 y_2 \dots y_N]^T$ is the received symbol from all the receive antenna,

$h = [h_1 h_2 \dots h_N]^T$ is the channel on all the receive antenna,

x is the transmitted symbol, and

$\eta = [\eta_1 \eta_2 \dots \eta_N]^T$ is the additive white Gaussian noise (AWGN) on all the receive antenna.

Effective $\frac{E_b}{N_0}$ with N receive antennas is N times $\frac{E_b}{N_0}$ for single antenna. Thus, the effective $\frac{E_b}{N_0}$ for N antennas in AWGN can be expressed as

$$\left[\frac{E_b}{N_0} \right]_{\text{eff},N} = \frac{NE_b}{N_0} \quad (2.29)$$

So the BER for N receive antennas is given by

$$P_b = \frac{1}{2} \operatorname{erfc} \left(\sqrt{\frac{NE_b}{N_0}} \right) \quad (2.30)$$

2.4 Diversity Combining Techniques

The three main combining techniques that can be used in conjunction with any of the diversity schemes are as follows:

1. Selection combining
2. Equal gain combining
3. Maximal ratio combining.

2.4.1 Selection Diversity

In this combiner, the receiver selects the antenna with the highest received signal power and ignores observations from the other antennas.

2.4.1.1 Expression for BER with Selection Diversity

Consider N independent Rayleigh fading channels each channel being a diversity branch. It is assumed that each branch has the same average signal-to-noise ratio

$$\bar{\gamma} = \frac{E_b}{N_0} E[h^2] \quad (2.31)$$

The outage probability is the probability that the bit energy to noise ratio falls below a threshold (γ). The probability of outage on i th receive antenna can be expressed by

$$P_{\text{out},\gamma_i} = P[\gamma_i < \gamma] = \int_0^\gamma \frac{1}{\bar{\gamma}} e^{-\frac{\gamma_i}{\bar{\gamma}}} d\gamma_i = 1 - e^{-\frac{\gamma_i}{\bar{\gamma}}} \quad (2.32)$$

The joint probability is the product of the individual probabilities if the channel on each antenna is assumed to be independent; thus, the joint probability with N receiving antennas becomes

$$\begin{aligned} P_{\text{out}} &= P[\gamma_1 < \gamma]P[\gamma_2 < \gamma] \dots P[\gamma_N < \gamma] = \prod_{i=1}^N P[\gamma_i < \gamma] \\ &= \left[1 - e^{-\frac{\gamma_i}{\bar{\gamma}}}\right]^N \end{aligned} \quad (2.33)$$

where $\gamma_1, \gamma_2, \dots, \gamma_N$ are the instantaneous bit energy to noise ratios of the first, second, and so on till the n th receive antenna.

Equation (2.33) is in fact the cumulative distribution function (CDF) of γ . Then, the probability density function (PDF) is given by the derivate of the CDF as

$$P(\gamma) = \frac{dP_{\text{out}}}{d\gamma} = \frac{N}{\bar{\gamma}} e^{-\frac{\gamma_i}{\bar{\gamma}}} \left[1 - e^{-\frac{\gamma_i}{\bar{\gamma}}}\right]^{N-1} \quad (2.34)$$

Substituting Eq. (2.34) in Eq. (2.1), BER for selective diversity can be expressed by

$$\text{BER}_{\text{SEL}} = \int_0^{\infty} \frac{1}{2} \operatorname{erfc}(\sqrt{\gamma}) \frac{N}{\bar{\gamma}} e^{-\frac{\gamma_i}{\bar{\gamma}}} \left[1 - e^{-\frac{\gamma_i}{\bar{\gamma}}}\right]^{N-1} d\gamma \quad (2.35)$$

Assuming $E[h^2] = 1$, the above expression can be rewritten as [12]

$$\text{BER}_{\text{SEL}} = \frac{1}{2} \sum_{k=0}^N (-1)^k \binom{N}{k} \left(1 + \frac{k}{\left(\frac{E_b}{N_0}\right)}\right)^{-\frac{1}{2}} \quad (2.36)$$

2.4.2 Equal Gain Combining (EGC)

In equal gain combining, equalization is performed on the i th receive antenna at the receiver by dividing the received symbol y_i by the a priori known phase of channel h_i . $|h_i|e^{j\theta_i}$ represents the channel h_i in polar form. The decoded symbol is obtained by

$$\hat{y} = \sum_i \frac{y_i}{e^{j\theta_i}} = \sum_i \frac{|h_i|e^{j\theta_i}x + \eta_i}{e^{j\theta_i}} = \sum_i |h_i|x + \tilde{\eta}_i \quad (2.37)$$

where

\hat{y} is the sum of the phase compensated channel from all the receiving antennas, and $\tilde{\eta}_i = \frac{\eta_i}{e^{j\theta_i}}$ is the additive noise scaled by the phase of the channel coefficient.

2.4.2.1 Expression for BER with Equal Gain Combining

The BER with equal gain combining with two receive antennas can be expressed with BPSK and BFSK modulations as [13]

$$\text{BER}_{\text{EGC,BPSK}} = \frac{1}{2} \left[1 - \frac{\sqrt{E_b/N_0(E_b/N_0 + 2)}}{E_b/N_0 + 1} \right] \quad (2.38)$$

$$\text{BER}_{\text{EGC,BFSK}} = \frac{1}{2} \left[1 - \frac{\sqrt{E_b/N_0(E_b/N_0 + 4)}}{E_b/N_0 + 2} \right] \quad (2.39)$$

2.4.3 Maximum Ratio Combining (MRC)

2.4.3.1 Expression for BER with Maximal Ratio Combining (MRC)

For channel h_i , the instantaneous bit energy to noise ratio at i th receive antenna is given by

$$\gamma_i = \frac{|h_i|^2 E_b}{N_0}, \quad (2.40)$$

If h_i is a Rayleigh-distributed random variable, then h_i^2 is a chi-squared random variable with two degrees of freedom. Hence, the pdf of γ_i can be expressed as

$$P_{\text{df}}(\gamma_i) = \frac{1}{(E_b/N_0)} e^{\frac{-\gamma_i}{(E_b/N_0)}} \quad (2.41)$$

Since the effective bit energy to noise ratio γ is the sum of N such random variables, the pdf of γ is a chi-square random variable with $2N$ degrees of freedom. Thus, the pdf of γ is given by

$$P_{\text{df}}(\gamma) = \frac{1}{(N-1)!(E_b/N_0)^N} \gamma^{N-1} e^{\frac{-\gamma}{(E_b/N_0)}}, \quad \gamma \geq 0 \quad (2.42)$$

Substituting Eq. (2.42) in Eq. (2.1), BER for maximal ratio combining can be expressed by

$$\text{BER}_{\text{MRC}} = \int_0^{\infty} \frac{1}{2} \operatorname{erfc}(\sqrt{\gamma}) P_{\text{df}}(\gamma) d\gamma$$

$$= \int_0^\infty \frac{1}{2} \operatorname{erfc}(\sqrt{\gamma}) \frac{1}{(N-1)!(E_b/N_0)^N} \gamma^{N-1} e^{\frac{-\gamma}{(E_b/N_0)}} d\gamma \quad (2.43)$$

The above expression can be rewritten [12] as

$$\text{BER}_{\text{MRC}} = P^N \sum_{k=0}^{N-1} \binom{N-1+k}{k} (1-P)^k \quad (2.44)$$

where

$$P = \frac{1}{2} - \frac{1}{2} \left(1 + \frac{1}{E_b/N_0} \right)^{-1/2}$$

The following MATLAB program computes the theoretic BER for BPSK modulation in Rayleigh fading channels with selective diversity, EGC, and MRC.

Program 2.7: Program for computing the theoretic BER for BPSK modulation in a Rayleigh fading channel with selection diversity, EGC, and MRC

```
clear all;
Eb_N0_dB = [0:20]; % multiple Eb/N0 values
EbN0Lin = 10.^((Eb_N0_dB/10));
theoryBer_nRx1 = 0.5.*((1-1*(1+1./EbN0Lin).^(0.5)));
theoryBer_sel_nRx2 = 0.5.*((1-2*(1+1./EbN0Lin).^(0.5)) + (1+2./EbN0Lin).^(0.5));
theoryBer_EG_nRx2 = 0.5*(1- sqrt(EbN0Lin.*((EbN0Lin+2))./(EbN0Lin+1)));
p = 1/2 - 1/2*(1+1./EbN0Lin).^(0.5);
theoryBer_MRC_nRx2 = p.^2.*((1+2*(1-p)));
semilogy(Eb_N0_dB, theoryBer_nRx1,'*', 'LineWidth', 2);
hold on
semilogy(Eb_N0_dB, theoryBer_sel_nRx2, '-','LineWidth', 2);
semilogy(Eb_N0_dB, theoryBer_EG_nRx2, '-+','LineWidth', 2);
semilogy(Eb_N0_dB, theoryBer_MRC_nRx2, '-','LineWidth', 2);
axis([0 20 10^-5 0.5])
gridon
legend('Rayleigh','selection(nRx=2)', 'EGC(nRx=2)', 'MRC(nRx=2)');
xlabel('Eb/No, dB');
ylabel('BER');
```

The BER performance from the above program with two receive antennas is shown in Fig. 2.16. From Fig. 2.16, it is observed that the BER with MRC is better than selective diversity and EGC and outperforms the single antenna case.

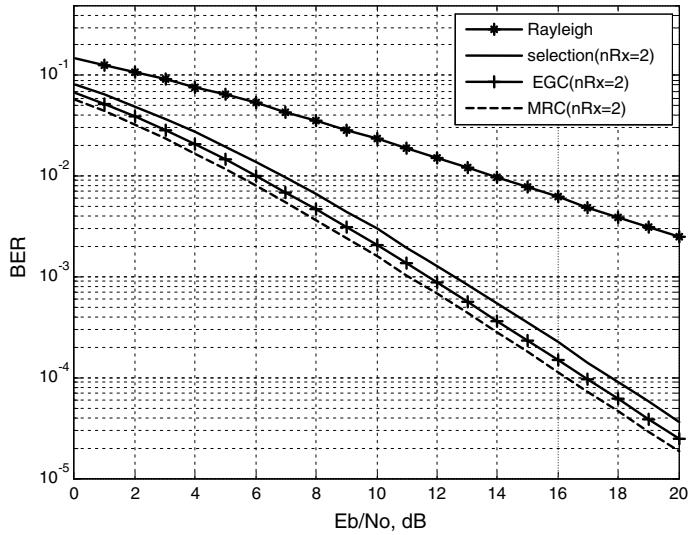


Fig. 2.16 Theoretic BER for BPSK modulation in a Rayleigh fading channel with selection diversity, EGC, and MRC

Example 2.1 What is the BER for $\frac{E_b}{N_0} = 8$ dB at the receiver output in an AWGN channel if coherently demodulated BPSK modulation is used and if no error control coding is used.

Solution Since $\frac{E_b}{N_0} > 6.5$ dB for BPSK modulation in AWGN channel, BER is given by

$$\text{BER}_{\text{BPSK}} = \frac{\exp\left(\frac{-E_b}{N_0}\right)}{\sqrt{4\pi\left(\frac{E_b}{N_0}\right)}}$$

$$\frac{E_b}{N_0} = 10^{(8/10)} = 6.3096$$

Thus,

$$\text{BER}_{\text{BPSK,AWGN}} = \frac{\exp(-6.3096)}{\sqrt{4\pi(6.3096)}} = 0.00020425.$$

Example 2.2 Determine $\frac{E_b}{N_0}$ required to achieve BER of 10^{-5} in a Rayleigh fading channel using BPSK

Solution BER for Rayleigh fading channel is given by

$$0.00001 = \frac{1}{2} \left(1 - \sqrt{\frac{E_b/N_0}{1 + E_b/N_0}} \right)$$

$$\frac{E_b}{N_0} = 25000.25; \frac{E_b}{N_0} (\text{dB}) = 10 \log 10(25000.25) = 44.$$

Example 2.3 Determine the necessary $\frac{E_b}{N_0}$ for a Rayleigh fading channel with an average BER of 10^{-5} in order to detect BFSK.

Solution BER for non-coherent BFSK in slow flat fading Rayleigh channel can be written as

$$0.00001 = \frac{1}{2 + \frac{E_b}{N_0}}$$

$$\frac{E_b}{N_0} = 99998; \frac{E_b}{N_0} (\text{dB}) = 10 \log 10(99998) = 50.$$

Example 2.4 Determine the necessary $\frac{E_b}{N_0}$ in order to detect BFSK with an average BER of 10^{-4} for a Rician fading channel with Rician factor of 5.

Solution BER for non-coherent BFSK in slow flat fading Rician channel is expressed by

$$0.0001 = \frac{1 + K}{2 + 2K + \frac{E_b}{N_0}} \exp\left(-\frac{K \frac{E_b}{N_0}}{2 + 2K + \frac{E_b}{N_0}}\right)$$

where K is the power ratio between the LOS path and non-LOS paths in the Rician fading channel.

For $K = 5$, BER becomes

$$0.0001 = \frac{1 + 5}{2 + 10 + \frac{E_b}{N_0}} \exp\left(-\frac{5 \frac{E_b}{N_0}}{2 + 10 + \frac{E_b}{N_0}}\right)$$

$$\text{Let } x = 10 + 2 + \frac{E_b}{N_0}; \frac{E_b}{N_0} = x - 2 - 10$$

$$10^{-4} = \frac{6}{x} e^{-5(\frac{x-2-10}{x})}$$

$$\left(\frac{10^{-4}}{6}x\right) = e^{-5(\frac{x-2-10}{x})}$$

$$\ln\left(\frac{10^{-4}}{6}\right) + \ln(x) = -5\left(\frac{x-2-10}{x}\right)$$

$$x \ln\left(\frac{10^{-4}}{6}\right) + x \ln(x) = -5(x-2-10)$$

Taking derivative of both sides of the above equation, we obtain

$$\ln\left(\frac{10^{-4}}{6}\right) + \left(x\frac{1}{x} + \ln(x)\right) = -5$$

$$-11+1 + \ln(x) = 5; \ln(x) = 6; x = e^6 = 403$$

$$\frac{E_b}{N_o} = 403 - 12 = 391$$

$$\frac{E_b}{N_o} (\text{dB}) = 10 \log_{10}(391) = 25.9 \text{ dB}$$

which yields $\frac{E_b}{N_0} (\text{dB}) = 26.5$.

2.5 Problems

1. Determine the probability of error as a function of $\frac{E_b}{N_0}$ for 4-QAM. Plot $\frac{E_b}{N_0}$ versus probability of error and compare the results with BPSK and non-coherent BFSK on the same plot.
2. Obtain an approximations to the outage capacity in a Rayleigh fading channel
 - (i) at low SNRs
 - (ii) at high SNRs.
3. Obtain an approximation to the outage probability for the parallel channel with M Rayleigh branches.
4. Assuming three-branch MRC diversity in a Rayleigh fading channel. For an average SNR of 20 dB, determine the outage probability that the SNR is below 10 dB.

2.6 MATLAB Exercises

1. Write a MATLAB program to simulate the BER versus number of users' performance of SFH-CDMA in AWGN and Rayleigh fading channels at different $\frac{E_b}{N_0}$.
2. Write a MATLAB program to simulate the performance of OFDM in AWGN and Rayleigh fading channels.
3. Write a MATLAB program to simulate the BER versus number of users performance of MC-CDMA in AWGN and Rayleigh fading channels for different number of subcarriers at different $\frac{E_b}{N_0}$.

4. Write a MATLAB program to simulate the performance of selection diversity, equal gain combiner, and maximum ratio combiner and compare the performance with the theoretical results.

References

1. J. Lu, K.B. Lataief, J.C.I.- Chuang, M.L. Liou, M-PSK and M-QAM BER computation using single space concepts. *IEEE Trans. Commun.* **47**, 181–184 (1999)
2. J.G. Proakis, *Digital Communications*, 3rd edn. (McGraw-Hill, New York, 1995)
3. T.S. Rappaport, *Wireless Communications: Principles and Practice* (IEEE Press, Piscataway, 1996)
4. W.C. Lindsey, Error probabilities for Rician fading multichannel reception of binary and n-ary signals. *IEEE Trans. Inf. Theory* **IT-10**(4), 333–350 (1964)
5. J. Lu, K.B. Lataief, J.C.-I. Chuang, M.L. Liou, M-PSK and M-QAM BER computation using a signal-Space concepts. *IEEE Trans. Commun.* **47**(2), 181–184 (1999)
6. M.K. Simon, M.-S. Alouini, *Digital Communication Over Fading Channels: A Unified Approach to Performance Analysis* (Wiley, New York, 2000)
7. J. Cheng, N.C. Beaulieu, Accurate DS-CDMA Bit-error probability calculation in Rayleigh fading. *IEEE Trans. Wirel. Commun.* **1**(1) (2002)
8. E.A. Geraniotis, M.B. Parsley, Error probabilities for slow- frequency-hopped spread-spectrum multiple-access communications over fading channels. *IEEE Trans. Commun.* **Com-30**(5) (1982)
9. L.-L. Yang, L. Hanzo, Overlapping M-ary frequency shift keying spread-spectrum multiple-access systems using random signature sequences. *IEEE Trans. Veh. Technol.* **48**(6) (1999)
10. J.G. Goh, S.V. Maric, The capacities of frequency-hopped code-division multiple-access channels. *IEEE Trans. Inf. Theory* **44**(3) (1998)
11. Q. Shi, M. Latva-Aho, Exact bit error rate calculations for synchronous MC-CDMA over a Rayleigh fading channel. *IEEE Commun. Lett.* **6**(7), 276–278 (2002)
12. J.R. Barry, E.A. Lee, D.G. Messerschmitt, *Digital Communication* (Kluwer Academic Publishers, Massachusetts, 2004)
13. Q.T. Zhang, Probability of error for equal-gain combiners over Rayleigh channels: some closed-form solutions. *IEEE Trans. Commun.* **45**(3), 270–273 (1997)

Chapter 3

Galois Field Theory



A small portion of linear algebra and combinatorics are used in the development of Hamming codes, the first-generation error control codes. The design of error control codes such as BCH codes and Reed–Solomon codes relies on the structures of Galois fields and polynomials over Galois fields. This chapter presents briefly algebraic tools for understanding of Galois field theory used in error-correcting codes design.

3.1 Set

A set is defined as an arbitrary collection of objects or elements. The presence of an element X in the set S is denoted by $X \in S$ and if X is certainly not in S , it is denoted by $X \notin S$. An empty set contains zero elements. A set Y is called a subset of a set X if and only if every element of Y is in X . Y is a subset of X is often denoted by $Y \subset X$ which reads “ Y is containing in X .”

Consider two sets, S_1 and S_2 , the new set $S_1 \cup S_2$ is called the union of S_1 and S_2 having the elements in either S_1 or S_2 , or both. Another set $S_1 \cap S_2$ is called the intersection of S_1 and S_2 having the common elements in S_1 and S_2 . If the intersection of two sets is empty, they are said to be disjoint.

3.2 Group

A group is a set on which a binary multiplication operation “.” is defined such that the following requirements satisfied

1. For any elements a and b in \mathbf{G} , $a \cdot b$ is an element in \mathbf{G} .
2. For any elements a , b , and c in \mathbf{G} , the following associative law,

$$a \cdot (b \cdot c) = (a \cdot b) \cdot c$$

3. There is an element e in \mathbf{G} such that for every a in \mathbf{G} ,

$$a \cdot e = e \cdot a = a \quad (\text{identity})$$

4. For any elements a in \mathbf{G} , There is an element a^{-1} in \mathbf{G} such that,

$$a \cdot a^{-1} = a^{-1} \cdot a = e \quad (\text{inverse}).$$

3.3 Field

If the addition and multiplication operations are defined on a set of objects F , then F is said to be a field if and only if

1. F forms a commutative group under addition.
2. F forms a commutative group under multiplication.
3. Addition and multiplications are distributive.

$$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$

The elements $F = \{1, 2, 3, \dots, p - 1\}$ forms a commutative group of order $(p - 1)$ under modulo p multiplication if and only if p is a prime integer.

All elements of a field form an additive commutative group, whereas all the nonzero elements of a field form a multiplicative commutative group. It is very useful to construct the finite fields. A Galois field which is particularly interesting to the coding theory is a field of finite order.

A Galois field of order q is usually denoted by $\text{GF}(q)$. The simplest of the Galois fields is $\text{GF}(2)$. $\text{GF}(2)$ can be represented by the two-element set $\{0, 1\}$ under standard binary addition and multiplication. The modulo-2 addition and multiplication are shown in Table 3.1.

Galois fields of size p , p a prime, can be constructed by modulo addition and multiplication. If these two operations are allowed to distribute, then a field is formed. The integers $\{0, 1, 2, \dots, p - 1\}$, form the field $\text{GF}(p)$ under modulo p addition and

Table 3.1 Addition and multiplication for $\text{GF}(2)$

Modulo-2 addition			Modulo-2 multiplication		
+	0	1	.	0	1
0	0	1	0	0	0
1	1	0	1	0	1

multiplication. The field GF(3) has the elements {0, 1, 2}. Finite fields GF(q) do not exist for all values of q . The value of q must be equal to p^m , where p is a prime positive integer and m is a positive integer. The finite fields of order p^m can be constructed as vector spaces over the prime order field GF(p).

Example 3.1 Construct addition and multiplication tables over GF(7).

Solution Here, p equals to 7; therefore, the elements of the GF are (0, 1, 2, 3, 4, 5, 6). The addition and multiplication over GF(7) will be modulo-7 as shown in Table 3.2.

3.4 Vector Spaces

Let V be a set of vector elements on which a binary addition operation is defined. Let F be a field with scalar elements, and a scalar multiplication operation is defined between the elements of V and the scalar elements of $F \cdot V$ forms a vector space over F if the following properties are satisfied.

Vector spaces

1. V is a commutative group under addition operation on V .
2. For any element $a \in F$ and any element $v \in V$, $a \cdot v \in V$.
3. For any elements $a, b \in F$ and any element $v \in V$, the following associativity law is satisfied

$$(a \cdot b) \cdot v = a \cdot (b \cdot v). \quad (3.1)$$

4. For any elements $a, b \in F$ and any elements $u, v \in V$, the following distributive law is satisfied

$$a \cdot (u + v) = a \cdot u + a \cdot v \quad (3.2)$$

$$(a + b) \cdot v = a \cdot v + b \cdot v \quad (3.3)$$

5. If 1 is the unit element $\in F$, for any element $v \in V$, $1 \cdot v = v$.

In the case of vector spaces over the scalar field GF(2), V is a collection of binary n -tuples such that if $v_1, v_2 \in V$, then $v_1 + v_2 \in V$, where $+$ stands for componentwise EXCLUSIVE-OR operation. If $v_1 = v_2, 0 \in V$.

Theorem 3.1 *Let v_1, v_2, \dots, v_k is a set of vectors in a vector space V over a finite field F with dimension k scalars, there is a unique representation of every vector v in V is*

$$v = a_1 v_1 + a_2 v_2 + \dots + a_k v_k \quad (3.4)$$

Table 3.2 Addition and multiplication for GF(7)

Modulo-7 addition							Modulo-7 multiplication								
+	0	1	2	3	4	5	6	.	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6	0	0	0	0	0	0	0	0
1	1	2	3	4	5	6	0	1	0	1	2	3	4	5	6
2	2	3	4	5	6	0	1	2	0	2	4	6	1	3	5
3	3	4	5	6	0	1	2	3	0	3	6	2	5	1	4
4	4	5	6	0	1	2	3	4	0	4	1	5	2	6	3
5	5	6	0	1	2	3	4	5	0	5	3	1	6	4	2
6	6	0	1	2	3	4	5	6	0	6	5	4	3	2	1

3.5 Elementary Properties of Galois Fields

1. Let α be an element in $\text{GF}(q)$. The order of α is the smallest positive integer n such that $\alpha^n = 1$.
2. The order q of a Galois field $\text{GF}(q)$ must be a power of a prime.
3. Every $\text{GF}(q)$ has at least one element α of order $(q - 1)$, which is called a primitive element and that exists in a $\text{GF}(q)$ such that $\alpha^{(q-1)} = 1$.
4. All nonzero elements in $\text{GF}(q)$ are represented by the $(q - 1)$ consecutive powers of a primitive element α .
5. Let α be a nonzero element in a Galois field $\text{GF}(q)$ and n be the order of α , then n divides $q - 1$.

3.6 Galois Field Arithmetic

Finite field arithmetic is different from standard integer arithmetic. In finite field arithmetic, all operations are performed on limited number of elements and result in an element within the same field.

3.6.1 Addition and Subtraction of Polynomials

In standard integer arithmetic, addition and subtraction of polynomials are performed by adding or subtracting together, whereas in finite field, addition and subtraction are accomplished using the XOR operator and they are identical.

Example 3.2 Add the polynomials $(x^6 + x^4 + x + 1)$ and $(x^7 + x^6 + x^3 + x)$ in $\text{GF}(2)$.

Solution $(x^6 + x^4 + x + 1) + (x^7 + x^6 + x^3 + x) = x^7 + x^4 + x^3 + 1$.

The normal algebraic sum and the modulo-2 finite field sum of a few polynomials are tabulated in Table 3.3.

Table 3.3 Computation of polynomials in normal algebra and Galois field

p_1	p_2	$p_1 + p_2$ (normal algebra)	$p_1 + p_2$ (GF)
$x^3 + x^2 + x + 1$	$x^3 + x^2$	$2x^3 + 2x^2 + x + 1$	$x + 1$
$x^4 + x^3 + x^2$	$x^5 + x^2$	$x^5 + x^4 + x^3 + 2x^2$	$x^5 + x^4 + x^3$
$x^2 + 1$	$x^3 + 1$	$x^3 + x^2 + 2$	$x^3 + x^2$

3.6.2 Multiplication of Polynomials

Multiplication of polynomials in Galois field is same as integer arithmetic, but the addition performed after multiplication is similar to Galois field.

Example 3.3 Multiply the polynomials $(x^6 + x^4 + x + 1)$ and $(x^7 + x^6 + x^3 + x)$.

Solution

$$\begin{aligned} & (x^6 + x^4 + x + 1)(x^7 + x^6 + x^3 + x) \\ &= x^{13} + x^{12} + x^9 + x^7 + x^{11} + x^{10} + x^7 + x^5 + x^8 + x^7 + x^4 + x^2 + x^7 + x^6 \\ &= +x^3 + x^{13} + x^{12} + x^9 + x^{11} + x^{10} + x^5 + x^8 + x^4 + x^2 + x^6 + x^3 + x \\ &= x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + x^2 + x \end{aligned}$$

3.6.3 Multiplication of Polynomials Using MATLAB

The following MATLAB command computes the multiplication of polynomial p_1 and polynomial p_2 in GF(2).

$$p_3 = \text{gfconv}(p_1, p_2)$$

The degree of the resulting GF(2) polynomial p_3 equals the sum of degree of the polynomial p_1 and degree of the polynomial p_2 .

For example, the following commands result the multiplication of the polynomials $1 + x + x^3$ and $1 + x + x^2 + x^4$.

$$\begin{aligned} p_1 &= [1 1 0 1]\%1 + x + x^3 \\ p_2 &= [1 1 1 0 1]\%1 + x + x^2 + x^4 \\ p_3 &= \text{gfconv}(p_1, p_2); \% (1 + x + x^3) * (1 + x + x^2 + x^4) \end{aligned}$$

The output p_3 for the above commands is

$$p_3 = [1 0 0 0 0 0 0 1]\%1 + x^7$$

3.6.4 Division of Polynomials

Suppose that $a(x)$ and $b(x) \neq 0$ are polynomials over GF(2). There are unique pair of polynomial called the quotient and remainder, $q(x)$ and $r(x)$ over GF(2), such that

$$a(x) = q(x)b(x) + r(x) \quad (3.5)$$

Example 3.4 Divide $f_1(x) = 1 + x^2 + x^3 + x^5$ by $f_2(x) = 1 + x^3 + x^4$.

Solution

$$\begin{array}{r} 1 + x \text{(quotient)} \\ 1 + x^3 + x^4) \overline{1 + x^2 + x^3 + x^5} \\ 1 \qquad + x^3 + x^4 \\ \hline x^2 \qquad + x^4 + x^5 \\ x \qquad + x^4 + x^5 \\ \hline \text{remainder: } x + x^2 \end{array}$$

It can easily be verified that

$$1 + x^2 + x^3 + x^5 = (1 + x^3 + x^4)(1 + x) + (x + x^2).$$

If the remainder $r(x)$ is zero, $a(x)$ is divisible by $b(x)$ and $b(x)$ is a factor of $a(x)$.

Example 3.5 Check whether $f_1(x) = (x^2 + x + 1)$ is a factor of $f_2(x) = x^5 + x^4 + 1$.

Solution

$$\begin{array}{r} x^3 + x + 1 \text{(quotient)} \\ x^2 + x + 1) \overline{x^5 + x^4 + 1} \\ x^5 + x^4 + x^3 \\ \hline x^3 + 1 \\ x^3 + x^2 + x \\ \hline x^2 + x + 1 \\ x^2 + x + 1 \\ \hline \text{remainder : 0} \end{array}$$

The remainder is zero, and hence, $f_2(x)$ is divisible by $f_1(x)$, and $f_1(x)$ is a factor of $f_2(x)$.

3.6.5 Division of Polynomials Using MATLAB

The following MATLAB command computes the quotient q and remainder r of the division of polynomial p_2 by polynomial p_1 in GF(2).

$$[q, r] = gfddeconv(p_2, p_1)$$

For example, the following commands divide polynomial $1 + x^7$ by polynomial $1 + x + x^3$

$$\begin{aligned} p_1 &= [1 \ 1 \ 0 \ 1] \% 1 + x + x^3 \\ p_2 &= [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1] \% 1 + x^7 \\ [q, r] &= gfddeconv(p_2, p_1); \% 1 + x^7 / 1 + x + x^3 \end{aligned}$$

The output q and r for the above commands are

$$q = [1 \ 1 \ 1 \ 0 \ 1] \% 1 + x + x^2 + x^4$$

$$r = 0.$$

3.7 Polynomials Over Galois Fields

A polynomial over $\text{GF}(q)$ is of the following form

$$a_0 + a_1x + a_2x^2 + \cdots + a_nx^n \quad (3.6)$$

of degree n (with $a_n \neq 0$) and with coefficients $\{a_i\}$ in the finite field $\text{GF}(q)$.

3.7.1 Irreducible Polynomial

A polynomial $p(x)$ is said to be irreducible in $\text{GF}(q)$, if $p(x)$ has no divisor polynomials in $\text{GF}(q)$ of degree less than m but greater than zero.

Examples

1. $x^3 + x^2 + 1$ is irreducible in $\text{GF}(2)$ as it is not factorable having degree of less than 3.
2. $x^4 + x^2 + 1$ is not irreducible in $\text{GF}(2)$, since the polynomial is divisible by the polynomial $x^2 + x + 1$ with coefficients in $\text{GF}(2)$, and with degree of 2 less than 4.
3. $x^4 + x^3 + x^2 + x + 1$ is irreducible in $\text{GF}(2)$ as it is not factorable having factors of degree less than 4.
4. $x^5 + x^4 + 1$ is not irreducible in $\text{GF}(2)$. Since the polynomial is divisible by polynomials of degree less than 5.

3.7.2 Primitive Polynomials

An irreducible polynomial $p(x) \in \text{GF}(2)$ of degree m is said to be primitive if the smallest positive integer n for which $p(x)$ divides $x^n - 1$ is $n = 2^m - 1$.

The roots $\{\alpha_j\}$ of an m th-degree primitive polynomial $p(x) \in \text{GF}(2)$ have order $2^m - 1$. All primitive polynomials are irreducible polynomials, but all irreducible polynomials are not primitive.

Examples

1. $x^2 + x + 1$ is primitive. The smallest polynomial of the form $x^n - 1$ for which it is a divisor is $x^3 - 1$ ($3 = 2^2 - 1$).
2. $x^3 + x^2 + 1$ is primitive. The smallest polynomial of the form $x^n - 1$ for which it is a divisor is $x^7 - 1$ ($7 = 2^3 - 1$).
3. $x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$ is not primitive since it is not irreducible. It can be factorized as product of the polynomials $x^3 + x^2 + 1$ and $x^3 + x + 1$.

3.7.3 Checking of Polynomials for Primitiveness Using MATLAB

The following MATLAB command can be used to check whether the degree- m GF(2) polynomial p is primitive.

`ck = gfprimck(p);`

The output `ck` is as follows:

- `ck = -1` p is not an irreducible polynomial
- `ck = 0` p is irreducible but not a primitive polynomial
- `ck = 1` p is a primitive polynomial

For example, the following MATLAB commands determine whether the polynomial $p(x) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$ is primitive or not

$$p = [1 1 1 1 1 1 1] \% 1 + x + x^3 + x^3 + x^4 + x^5 + x^6$$

`ck = gfprimck(p);`

The output `ck` for the above commands is -1 indicating that the polynomial is not irreducible and hence not primitive.

3.7.4 Generation of Primitive Polynomials Using MATLAB

Primitive polynomials of degree m can be generated using the following MATLAB command `primpoly` as follows

$$p = \text{primpoly}(m, \text{'all'})$$

For example, the primitive polynomials generated using the above m file for $m = 3, 4, 5$, and 6 are tabulated in Table 3.4.

3.8 Construction of Galois Field $\text{GF}(2^m)$ from $\text{GF}(2)$

The 2^m elements of $\text{GF}(2^m)$ can be written as $\{0, 1, \alpha, \alpha^2, \alpha^3, \dots, \alpha^{2^m-2}\}$. All the nonzero elements of $\text{GF}(2^m)$ are generated by the powers of the primitive element α that satisfies the condition $\alpha^{2^m-1} = 1$.

The polynomial representation of the elements of $\text{GF}(2^m)$ is given by the remainder of x^n upon division by the polynomial $p(x)$ which is primitive in $\text{GF}(2)$.

$$\alpha^n = \text{Remainder}\{x^n / \cdot p(x)\} \quad (3.7)$$

Table 3.4 Primitive polynomials for $m = 3, 4, 5$, and 6

m	Primitive polynomial $p(x)$
3	$p(x) = x^3 + x + 1$ $p(x) = x^3 + x^2 + 1$
4	$p(x) = x^4 + x + 1$ $p(x) = x^4 + x^3 + 1$
5	$p(x) = x^5 + x^2 + 1$ $p(x) = x^5 + x^3 + 1$ $p(x) = x^5 + x^3 + x^2 + x + 1$ $p(x) = x^5 + x^4 + x^2 + x + 1$ $p(x) = x^5 + x^4 + x^3 + x + 1$ $p(x) = x^5 + x^4 + x^3 + x^2 + 1$
6	$p(x) = x^6 + x + 1$ $p(x) = x^6 + x^4 + x^3 + x + 1$ $p(x) = x^6 + x^5 + 1$ $p(x) = x^6 + x^5 + x^2 + x + 1$ $p(x) = x^6 + x^5 + x^3 + x^2 + 1$ $p(x) = x^6 + x^5 + x^4 + x + 1$

Example 3.6 Construct GF(8) as a vector space over GF(2).

Solution Let us consider the construction of GF(8) based on the polynomial $p(x) = x^3 + x + 1$ which is primitive in GF(2). Let α be a root of $p(x)$. This implies that $\alpha^3 + \alpha + 1 = 0$, or equivalently $\alpha^3 = \alpha + 1$. The distinct powers of α must have $(2^3 - 1)$ distinct nonzero polynomial representations of α of degree 2 or less with the coefficients from GF(2). The set $\{1, \alpha, \alpha^2\}$ is used as a basis for the vector space representation of GF(8)

Since every field must contain zero and one element, we have:

$$\begin{aligned} 0 &= 0 \\ \alpha^0 &= 1 \end{aligned}$$

Since the reminders of x and x^2 upon division by the primitive polynomial $p(x) = x^3 + x + 1$ are themselves, the other two possible assignments are as follows:

$$\begin{aligned} \alpha^1 &= x \\ \alpha^2 &= x^2 \end{aligned}$$

However, the polynomial representation of x^3 can be derived by the following polynomial division:

$$\begin{array}{r} 1 \\ \hline x^3 + x + 1 \) x^3 \\ \underline{x^3 + x + 1} \\ -(x + 1) \end{array}$$

The remainder is $-(x + 1)$. Hence, $\alpha^3 = \alpha + 1$

For x^4

$$\begin{array}{r} x \\ \hline x^3 + x + 1 \) x^4 \\ \underline{x^4 + x^2 + x} \\ -(x^2 + x) \end{array}$$

The remainder is $-(x^2 + x)$. Hence, $\alpha^4 = \alpha^2 + \alpha$

For x^5

$$\begin{array}{r} x^2 + 1 \\ \hline x^3 + x + 1) x^5 \\ \underline{x^5 + x^3 + x^2} \\ -x^3 - x^2 \\ \underline{x^3 + x + 1} \\ -(x^2 + x + 1) \end{array}$$

The remainder is $-(x^2 + x + 1)$. Hence, $\alpha^5 = \alpha^2 + \alpha + 1$.

Similarly, for x^6 and x^7 , follow the same procedure to get the polynomial representations. All the above values are tabulated in the following tabulated Table 3.5.

Example 3.7 Construct GF(16) as a vector space over GF(2).

Solution Let us consider the construction of GF(16) based on the polynomial $p(x) = x^4 + x + 1$ which is primitive in GF(2).

We know that

$$0 = 0$$

$$\alpha^0 = 1$$

Since the reminders of x , x^2 and x^3 upon division by the primitive polynomial $p(x) = x^4 + x + 1$ are themselves, the other three possible assignments are as follows:

$$\alpha^1 = x$$

$$\alpha^2 = x^2$$

$$\alpha^3 = x^3$$

Table 3.5 Representation of elements of GF(8)

Zero and powers of α	Polynomial representation	Vector space over GF(2) $1 \alpha \alpha^2$
0	0	000
α^0	1	100
α^1	α	010
α^2	α^2	001
α^3	$1 + \alpha$	110
α^4	$\alpha + \alpha^2$	011
α^5	$1 + \alpha + \alpha^2$	111
α^6	$1 + \alpha^2$	101
α^7	1	100

However, the polynomial representation of x^4 can be derived by the following polynomial division:

$$\begin{array}{r} \underline{1} \\ x^4 + x + 1) \underline{x^4} \\ \underline{x^4 + x + 1} \\ -(x + 1) \end{array}$$

The remainder is $-(x + 1)$. Hence, $\alpha^4 = \alpha + 1$

Similarly, for the values from x^5 to x^{15} , follow the same procedure to get the polynomial representations. All the above values are tabulated in the following Table 3.6

Example 3.8 Construct GF(32) as a vector space over GF(2).

Solution Let us consider the construction of GF(32) based on the polynomial $p(x) = x^5 + x^2 + 1$ which is primitive in GF(2).

We know that

$$\begin{aligned} 0 &= 0 \\ \alpha^0 &= 1 \end{aligned}$$

Table 3.6 Representation of elements of GF(16)

Zero and powers of α	Polynomials over GF(2)	Vector space over GF(2) $1 \alpha \alpha^2 \alpha^3$
0	0	0 0 0 0
α^0	1	1 0 0 0
α^1	α	0 1 0 0
α^2	α^2	0 0 1 0
α^3	α^3	0 0 0 1
α^4	$1 + \alpha$	1 1 0 0
α^5	$\alpha + \alpha^2$	0 1 1 0
α^6	$\alpha^2 + \alpha^3$	0 0 1 1
α^7	$1 + \alpha + \alpha^3$	1 1 0 1
α^8	$1 + \alpha^2$	1 0 1 0
α^9	$\alpha + \alpha^3$	0 1 0 1
α^{10}	$1 + \alpha + \alpha^2$	1 1 1 0
α^{11}	$\alpha + \alpha^2 + \alpha^3$	0 1 1 1
α^{12}	$1 + \alpha + \alpha^2 + \alpha^3$	1 1 1 1
α^{13}	$1 + \alpha^2 + \alpha^3$	1 0 1 1
α^{14}	$1 + \alpha^3$	1 0 0 1
α^{15}	1	1 0 0 0

Since the reminders of x , x^2 , x^3 , and x^4 upon division by the primitive polynomial $p(x) = x^5 + x^2 + 1$ are themselves, the other four possible assignments are as follows:

$$\alpha^1 = x$$

$$\alpha^2 = x^2$$

$$\alpha^3 = x^3$$

$$\alpha^4 = x^4$$

However, the polynomial representation of x^5 can be derived by the following polynomial division:

$$\begin{array}{r} 1 \\ \hline x^5 + x^2 + 1) \overline{x^5} \\ \underline{x^5 + x^2 + 1} \\ -(x^2 + 1) \end{array}$$

The remainder is $-(x^2 + 1)$. Hence, $\alpha^5 = \alpha^2 + 1$

Similarly, for the values from x^6 to x^{31} , follow the same procedure to get the polynomial representations. All the above values are tabulated in the following Table 3.7.

Example 3.9 Construct GF(64) as a vector space over GF(2).

Solution Let us consider the construction of GF(64) based on the polynomial $p(x) = x^6 + x + 1$ which is primitive in GF(2).

We know that

$$0 = 0$$

$$\alpha^0 = 1$$

Since the reminders of x , x^2 , x^3 , x^4 , and x^5 upon division by the primitive polynomial $p(x) = x^6 + x + 1$ are themselves, the other five possible assignments are:

$$\alpha^1 = x$$

$$\alpha^2 = x^2$$

$$\alpha^3 = x^3$$

$$\alpha^4 = x^4$$

$$\alpha^5 = x^5$$

Table 3.7 Representation of elements of GF(32)

Zero and powers of α	Polynomials over GF(2)	Vector space over GF(2) $1 \alpha \alpha^2 \alpha^3 \alpha^4$
0	0	0 0 0 0 0
α^0	1	1 0 0 0 0
α^1	α	0 1 0 0 0
α^2	α^2	0 0 1 0 0
α^3	α^3	0 0 0 1 0
α^4	α^4	0 0 0 0 1
α^5	$1 + \alpha^2$	1 0 1 0 0
α^6	$\alpha + \alpha^3$	0 1 0 1 0
α^7	$\alpha^2 + \alpha^4$	0 0 1 0 1
α^8	$1 + \alpha^2 + \alpha^3$	1 0 1 1 0
α^9	$\alpha + \alpha^3 + \alpha^4$	0 1 0 1 1
α^{10}	$1 + \alpha^4$	1 0 0 0 1
α^{11}	$1 + \alpha + \alpha^2$	1 1 1 0 0
α^{12}	$\alpha + \alpha^2 + \alpha^3$	0 1 1 1 0
α^{13}	$\alpha^2 + \alpha^3 + \alpha^4$	0 0 1 1 1
α^{14}	$1 + \alpha^2 + \alpha^3 + \alpha^4$	1 0 1 1 1
α^{15}	$1 + \alpha + \alpha^2 + \alpha^3 + \alpha^4$	1 1 1 1 1
α^{16}	$1 + \alpha + \alpha^3 + \alpha^4$	1 1 0 1 1
α^{17}	$1 + \alpha + \alpha^4$	1 1 0 0 1
α^{18}	$1 + \alpha$	1 1 0 0 0
α^{19}	$\alpha + \alpha^2$	0 1 1 0 0
α^{20}	$\alpha^2 + \alpha^3$	0 0 1 1 0
α^{21}	$\alpha^3 + \alpha^4$	0 0 0 1 1
α^{22}	$1 + \alpha^2 + \alpha^4$	1 0 1 0 1
α^{23}	$1 + \alpha + \alpha^2 + \alpha^3$	1 1 1 1 0
α^{24}	$\alpha + \alpha^2 + \alpha^3 + \alpha^4$	0 1 1 1 1
α^{25}	$1 + \alpha^3 + \alpha^4$	1 0 0 1 1
α^{26}	$1 + \alpha + \alpha^2 + \alpha^4$	1 1 1 0 1
α^{27}	$1 + \alpha + \alpha^3$	1 1 0 1 0
α^{28}	$\alpha + \alpha^2 + \alpha^4$	0 1 1 0 1
α^{29}	$1 + \alpha^3$	1 0 0 1 0
α^{30}	$\alpha + \alpha^4$	0 1 0 0 1
α^{31}	1	1 0 0 0 0

However, the polynomial representation of x^6 can be derived by the following polynomial division:

$$\begin{array}{r} \underline{1} \\ x^6 + x + 1) \underline{x^6} \\ \underline{x^6 + x + 1} \\ -(x + 1) \end{array}$$

The remainder is $-(x + 1)$. Hence, $\alpha^6 = \alpha + 1$

Similarly, for the values from x^7 to x^{63} , follow the same procedure to get the polynomial representations. All the above values are tabulated in the following Table 3.8.

3.8.1 Construction of $GF(2^m)$, Using MATLAB

To construct $GF(2^m)$, the following MATLAB function can be used

```
field = gftuple([-1:2^m-2]', m, 2);
```

For example, the $GF(8)$ generated using the above m file for $m=3$ is as follows

```
field =
```

0	0	0
1	0	0
0	1	0
0	0	1
1	1	0
0	1	1
1	1	1
1	0	1

3.9 Minimal Polynomials and Conjugacy Classes of $GF(2^m)$

3.9.1 Minimal Polynomials

Definition 3.1 Suppose that α is an element in $GF(2^m)$. The unique minimal polynomial of α with respect to $GF(2)$ is a polynomial $\phi(x)$ of minimum degree such that $\phi(\alpha) = 0$.

Table 3.8 Representation of elements of GF(64)

Zero and powers of α	Polynomials over GF(2)	Vector space over GF(2) $1 \alpha \alpha^2 \alpha^3 \alpha^4 \alpha^5$
0	1	0 0 0 0 0
α^0	1	1 0 0 0 0
α^1	α	0 1 0 0 0
α^2	α^2	0 0 1 0 0
α^3	α^3	0 0 0 1 0
α^4	α^4	0 0 0 0 1
α^5	α^5	0 0 0 0 0 1
α^6	$1 + \alpha$	1 1 0 0 0
α^7	$\alpha + \alpha^2$	0 1 1 0 0
α^8	$\alpha^2 + \alpha^3$	0 0 1 1 0
α^9	$\alpha^3 + \alpha^4$	0 0 0 1 1 0
α^{10}	$\alpha^4 + \alpha^5$	0 0 0 0 1 1
α^{11}	$1 + \alpha + \alpha^5$	1 1 0 0 0 1
α^{12}	$1 + \alpha^2$	1 0 1 0 0
α^{13}	$\alpha + \alpha^3$	0 1 0 1 0 0
α^{14}	$\alpha^2 + \alpha^4$	0 0 1 0 1 0
α^{15}	$\alpha^3 + \alpha^5$	0 0 0 1 0 1
α^{16}	$1 + \alpha + \alpha^4$	1 1 0 0 1 0
α^{17}	$\alpha + \alpha^2 + \alpha^5$	0 1 1 0 0 1
α^{18}	$1 + \alpha + \alpha^2 + \alpha^3$	1 1 1 1 0 0
α^{19}	$\alpha + \alpha^2 + \alpha^3 + \alpha^4$	0 1 1 1 1 0
α^{20}	$\alpha^2 + \alpha^3 + \alpha^4 + \alpha^5$	0 0 1 1 1 1
α^{21}	$1 + \alpha + \alpha^3 + \alpha^4 + \alpha^5$	1 1 0 1 1 1
α^{22}	$1 + \alpha^2 + \alpha^4 + \alpha^5$	1 0 1 0 1 1
α^{23}	$1 + \alpha^3 + \alpha^5$	1 0 0 1 0 1
α^{24}	$1 + \alpha^4$	1 0 0 0 1 0
α^{25}	$\alpha + \alpha^5$	0 1 0 0 0 1
α^{26}	$1 + \alpha + \alpha^2$	1 1 1 0 0 0
α^{27}	$\alpha + \alpha^2 + \alpha^3$	0 1 1 1 0 0
α^{28}	$\alpha^2 + \alpha^3 + \alpha^4$	0 0 1 1 1 0
α^{29}	$\alpha^3 + \alpha^4 + \alpha^5$	0 0 0 1 1 1
α^{30}	$1 + \alpha + \alpha^4 + \alpha^5$	1 1 0 0 1 1
α^{31}	$1 + \alpha^2 + \alpha^5$	1 0 1 0 0 1
α^{32}	$1 + \alpha^3$	1 0 0 1 0 0
α^{33}	$\alpha + \alpha^4$	0 1 0 0 1 0

(continued)

Table 3.8 (continued)

Zero and powers of α	Polynomials over GF(2)	Vector space over GF(2) $1 \alpha \alpha^2 \alpha^3 \alpha^4 \alpha^5$
α^{34}	$\alpha^2 + \alpha^5$	0 0 1 0 0 1
α^{35}	$1 + \alpha + \alpha^3$	1 1 0 1 0 0
α^{36}	$\alpha + \alpha^2 + \alpha^4$	0 1 1 0 1 0
α^{37}	$\alpha^2 + \alpha^3 + \alpha^5$	0 0 1 1 0 1
α^{38}	$1 + \alpha + \alpha^3 + \alpha^4$	1 1 0 1 1 0
α^{39}	$\alpha + \alpha^2 + \alpha^4 + \alpha^5$	0 1 1 0 1 1
α^{40}	$1 + \alpha + \alpha^2 + \alpha^3 + \alpha^5$	1 1 1 1 0 1
α^{41}	$1 + \alpha^2 + \alpha^3 + \alpha^4$	1 0 1 1 1 0
α^{42}	$\alpha + \alpha^3 + \alpha^4 + \alpha^5$	0 1 0 1 1 1
α^{43}	$1 + \alpha + \alpha^2 + \alpha^4 + \alpha^5$	1 1 1 0 1 1
α^{44}	$1 + \alpha^2 + \alpha^3 + \alpha^5$	1 0 1 1 0 1
α^{45}	$1 + \alpha^3 + \alpha^4$	1 0 0 1 1 0
α^{46}	$\alpha + \alpha^4 + \alpha^5$	0 1 0 0 1 1
α^{47}	$1 + \alpha + \alpha^2 + \alpha^5$	1 1 1 0 0 1
α^{48}	$1 + \alpha^2 + \alpha^3$	1 0 1 1 0 0
α^{49}	$\alpha + \alpha^3 + \alpha^4$	0 1 0 1 1 0
α^{50}	$\alpha^2 + \alpha^4 + \alpha^5$	0 0 1 0 1 1
α^{51}	$1 + \alpha + \alpha^3 + \alpha^5$	1 1 0 1 0 1
α^{52}	$1 + \alpha^2 + \alpha^4$	1 0 1 0 1 0
α^{53}	$\alpha + \alpha^3 + \alpha^5$	0 1 0 1 0 1
α^{54}	$1 + \alpha + \alpha^2 + \alpha^4$	1 1 1 0 1 0
α^{55}	$\alpha + \alpha^2 + \alpha^3 + \alpha^5$	0 1 1 1 0 1
α^{56}	$1 + \alpha + \alpha^2 + \alpha^3 + \alpha^4$	1 1 1 1 1 0
α^{57}	$\alpha + \alpha^2 + \alpha^3 + \alpha^4 + \alpha^5$	0 1 1 1 1 1
α^{58}	$1 + \alpha + \alpha^2 + \alpha^3 + \alpha^4 + \alpha^5$	1 1 1 1 1 1
α^{59}	$1 + \alpha^2 + \alpha^3 + \alpha^4 + \alpha^5$	1 0 1 1 1 1
α^{60}	$1 + \alpha^3 + \alpha^4 + \alpha^5$	1 0 0 1 1 1
α^{61}	$1 + \alpha^4 + \alpha^5$	1 0 0 0 1 1
α^{62}	$1 + \alpha^5$	1 0 0 0 0 1
α^{63}	1	1 0 0 0 0 0

3.9.2 Conjugates of GF Elements

Let α be an element in the Galois field GF(2^m). The conjugates of α with respect to the subfield GF(q) are the elements $\alpha, \alpha^2, \alpha^{2^2}, \alpha^{2^3}, \dots$

The conjugates of α with respect to GF(q) form a set called the conjugacy class of α with respect to GF(q).

Theorem 3.2 Conjugacy Class *The conjugacy class of $\alpha \in GF(2^m)$ with respect to GF(2) contains all the elements of the form α^{2^i} for $0 \leq i \leq l - 1$ where l is the smallest positive integer such that $\alpha^{2^l} = \alpha$.*

3.9.3 Properties of Minimal Polynomial

Theorem 3.3 *The minimal polynomial of an element of α of GF(2^m) is an irreducible polynomial.*

Proof Suppose the minimal polynomial $\phi(x)$ is not irreducible.

Then $\phi(x)$ can be expressed as a product of two other polynomials

$$\phi(x) = \phi_1(x)\phi_2(x)$$

As $\phi(\alpha) = \phi_1(\alpha)\phi_2(\alpha) = 0$, either $\phi_1(\alpha) = 0$ or $\phi_2(\alpha) = 0$.

It is contradictory with the minimality of the degree $\phi(x)$.

Theorem 3.4 *Let $f(x)$ be a polynomial over GF(2) and $\phi(x)$ be the minimal polynomial of an element α in GF(2^m). If α is a root of $f(x)$, then $f(x)$ is divisible by $\phi(x)$.*

Proof The division of $f(x)$ by $\phi(x)$ gives

$$f(x) = \phi(x)q(x) + r(x)$$

Since α is a root of $f(x)$, $f(\alpha) = 0$ and $\phi(\alpha) = 0$, it follows that $r(\alpha) = 0$. As the degree of $r(x)$ is less than that of $\phi(x)$, $r(\alpha) = 0$ only when $r(x) = 0$. Hence, $f(x) = \phi(x)q(x)$; therefore, $f(x)$ is divisible by $\phi(x)$.

Theorem 3.5 *The nonzero elements of GF(2^m) form all the roots of $x^{2^m-1} - 1$.*

Proof Let α be a nonzero elements in the field GF(2^m). Then, it follows that $\alpha^{2^m-1} = 1$, or $\alpha^{2^m-1} + 1 = 0$. This implies that α is a root of the polynomial $x^{2^m-1} + 1$. Hence, every nonzero element of GF(2^m) is a root of $x^{2^m-1} + 1$. Since the degree of $x^{2^m-1} + 1$ is $2^m - 1$, the $2^m - 1$ nonzero elements of GF(2^m) form all the roots of $x^{2^m-1} + 1$.

Theorem 3.6 Let α be an element in the Galois field $GF(2^m)$. Then, all its conjugates $\alpha, \alpha^2, \dots, \alpha^{2^{l-1}}$ have the same minimal polynomial.

A direct consequence of Theorem 3.5 is that $x^{2^m-1} - 1$ is equal to the product of the distinct minimal polynomials of the nonzero elements of $GF(2^m)$.

Theorem 3.7 Suppose that $\phi(x)$ be the minimal polynomial of an element α of $GF(2^m)$ and l be the smallest positive integer such that $\alpha^{2^l} = \alpha$ then $\phi(x)$ of degree m or less is given by

$$\phi(x) = \prod_{i=0}^{l-1} (x - \alpha^{2^i}) \quad (3.8)$$

3.9.4 Construction of Minimal Polynomials

The stepwise procedure for the construction of the Galois field is as follows

- Step 1. Generate the Galois field $GF(2^m)$ based on the primitive polynomial corresponding to m .
- Step 2. Find the groups of the conjugate roots.
- Step 3. The construction of minimal polynomial of each element is by using Eq. (3.8).

Using the above procedure, the following examples illustrate the construction of the minimal polynomial for $GF(8)$, $GF(16)$, and $GF(32)$ with respect to $GF(2)$.

Example 3.10 Determine the minimal polynomials of the elements of $GF(8)$ with respect to $GF(2)$.

Solution The eight elements in $GF(8)$ are arranged in conjugacy classes and their minimal polynomials computed as follows

Conjugacy class	Associated minimal polynomial
$\{0\}$	$\phi^*(x) = x$
$\{1\}$	$\phi_0(x) = x + 1$
$\{\alpha, \alpha^2, \alpha^4\}$	$\phi_1(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^4) = x^3 + x + 1$
$\{\alpha^3, \alpha^6, \alpha^5\}$	$\phi_3(x) = (x - \alpha^3)(x - \alpha^6)(x - \alpha^5) = x^3 + x^2 + 1$

From the Theorem 3.5, it is known that the minimal polynomials of the nonzero elements in the field $GF(8)$ provide the complete factorization of $x^7 - 1$. Hence,

$$x^7 - 1 = (x + 1)(x^3 + x + 1)(x^3 + x^2 + 1).$$

Example 3.11 Determine the minimal polynomials of the elements of $\text{GF}(16)$ with respect to $\text{GF}(2)$.

Solution The 16 elements in $\text{GF}(2^4)$ are arranged in conjugacy classes and their associated minimal polynomials computed as follows:

Conjugacy class	Associated minimal polynomial
$\{0\}$	$\phi^*(x) = x$
$\{1\}$	$\phi_0(x) = x + 1$
$\{\alpha, \alpha^2, \alpha^4, \alpha^8\}$	$\begin{aligned}\phi_1(x) &= (x - \alpha)(x - \alpha^2)(x - \alpha^4)(x - \alpha^8) \\ &= x^4 + x + 1\end{aligned}$
$\{\alpha^3, \alpha^6, \alpha^{12}, \alpha^9\}$	$\begin{aligned}\phi_3(x) &= (x - \alpha^3)(x - \alpha^6)(x - \alpha^{12})(x - \alpha^9) \\ &= x^4 + x^3 + x^2 + x + 1\end{aligned}$
$\{\alpha^5, \alpha^{10}\}$	$\phi_5(x) = (x - \alpha^5)(x - \alpha^{10}) = x^2 + x + 1$
$\{\alpha^7, \alpha^{14}, \alpha^{13}, \alpha^{11}\}$	$\begin{aligned}\phi_9(x) &= (x - \alpha^7)(x - \alpha^{14})(x - \alpha^{13})(x - \alpha^{11}) \\ &= x^4 + x^3 + 1\end{aligned}$

As a consequence of the Theorem 3.5, the following factorization holds good for $\text{GF}(16)$

$$x^{15} - 1 = (x + 1)(x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)(x^2 + x + 1)(x^4 + x^3 + 1)$$

Example 3.12 Determine the minimal polynomials of the elements of $\text{GF}(32)$ with respect to $\text{GF}(2)$.

Solution The 32 elements in $\text{GF}(32)$ are arranged in conjugacy classes and their minimal polynomials computed as follows

Conjugacy class	Associated minimal polynomial
$\{0\}$	$\phi^*(x) = x$
$\{1\}$	$\phi_0(x) = x + 1$
$\{\alpha, \alpha^2, \alpha^4, \alpha^8, \alpha^{16}\}$	$\begin{aligned}\phi_1(x) &= (x - \alpha)(x - \alpha^2)(x - \alpha^4)(x - \alpha^8)(x - \alpha^{16}) \\ &= x^5 + x^2 + 1\end{aligned}$
$\{\alpha^3, \alpha^6, \alpha^{12}, \alpha^{24}, \alpha^{17}\}$	$\begin{aligned}\phi_3(x) &= (x - \alpha^3)(x - \alpha^6)(x - \alpha^{12})(x - \alpha^{24})(x - \alpha^{17}) \\ &= x^5 + x^4 + x^3 + x^2 + 1\end{aligned}$
$\{\alpha^5, \alpha^{10}, \alpha^{20}, \alpha^9, \alpha^{18}\}$	$\begin{aligned}\phi_5(x) &= (x - \alpha^5)(x - \alpha^{10})(x - \alpha^{20})(x - \alpha^9)(x - \alpha^{18}) \\ &= x^5 + x^4 + x^2 + x + 1\end{aligned}$
$\{\alpha^7, \alpha^{14}, \alpha^{28}, \alpha^{25}, \alpha^{19}\}$	$\begin{aligned}\phi_9(x) &= (x - \alpha^7)(x - \alpha^{14})(x - \alpha^{28})(x - \alpha^{25})(x - \alpha^{19}) \\ &= x^5 + x^3 + x^2 + x + 1\end{aligned}$

(continued)

(continued)

Conjugacy class	Associated minimal polynomial
$\{\alpha^{11}, \alpha^{22}, \alpha^{13}, \alpha^{26}, \alpha^{21}\}$	$\begin{aligned}\phi_{11}(x) &= (x - \alpha^{11})(x - \alpha^{22})(x - \alpha^{13})(x - \alpha^{26})(x - \alpha^{21}) \\ &= x^5 + x^4 + x^3 + x + 1\end{aligned}$
$\{\alpha^{15}, \alpha^{30}, \alpha^{29}, \alpha^{27}, \alpha^{23}\}$	$\begin{aligned}\phi_{15}(x) &= (x - \alpha^{15})(x - \alpha^{30})(x - \alpha^{29})(x - \alpha^{27})(x - \alpha^{23}) \\ &= x^5 + x^3 + 1\end{aligned}$

According to the Theorem 3.5, the following factorization is valid for GF(32)

$$\begin{aligned}x^{31} - 1 &= (x + 1)(x^5 + x^2 + 1)(x^5 + x^4 + x^3 + x^2 + 1)(x^5 + x^4 + x^2 + x + 1) \\ &\quad (x^5 + x^3 + x^2 + x + 1)(x^5 + x^4 + x^3 + x + 1)(x^5 + x^3 + 1)\end{aligned}$$

3.9.5 Construction of Conjugacy Classes Using MATLAB

```
cst = cosets(m)
```

The matlab command can be used to find the Conjugacy Classes for the nonzero elements in GF (8).

For Example, for m=3, the Conjugacy Classes are generated using the above matlab command is as follows

```
c = cosets(3);
c{1}'
c{2}'
c{3}'
```

$c\{1\}'$ displays the conjugacy class $\{\alpha^0\}$ which indicates the non zero element α^0 .

$c\{2\}'$ displays the conjugacy class $\{\alpha, \alpha^2, \alpha^4\}$ which represents non zero elements α, α^2 and $\alpha^2 + \alpha$ respectively.

$c\{3\}'$ displays the conjugacy class $\{\alpha^3, \alpha^6, \alpha^5\}$ which represents non zero elements $\alpha + 1, 1 + \alpha^2$ and $\alpha^2 + \alpha + 1$ respectively

3.9.6 Construction of Minimal Polynomials Using MATLAB

The conjugacy classes of the elements of $GF(2^m)$ and associated minimal polynomials can be constructed using the MATLAB commands cosets and minpol. For example, for $GF(2^4)$, the following MATLAB program constructs the minimal polynomial of the conjugacy class in which α^7 is an element.

```

clear all;
clc;
m=4;
c = cosets(m); % computes conjugacy classes for GF(16)
c{5}'; % conjugacy class in which  $\alpha^7$  is an element
A=gf([c{5}'],m);
pol=minpol(A')% displays coefficients of the minimal
polynomial in descending order
The output of the above program is as follows
Primitive polynomial(s) =
x^4+x^1+1
pol = GF(2) array.
Array elements =
1     1     0     0     1
1     1     0     0     1
1     1     0     0     1
1     1     0     0     1

```

From the output, array elements indicate the coefficients minimal polynomials in the descending order for four elements in the conjugacy class. Hence, the minimal polynomial for the conjugacy class which α^7 is an element is given by $\phi(x) = x^4 + x^3 + 1$.

3.10 Problems

1. Construct modulo-5 addition and multiplication tables for GF(5)
2. Divide the polynomial $f(x) = 1 + x + x^4 + x^5 + x^6$ by the polynomial $g(x) = 1 + x + x^3$ in GF(2).
3. Find whether each of the following polynomial is irreducible in GF(2).
 - (a) $p(x) = x^2 + x + 1$
 - (b) $p(x) = x^{11} + x^2 + 1$
 - (c) $p(x) = x^{21} + x^2 + 1$
4. Find whether each of the following polynomial is primitive in GF(2).
 - (a) $p(x) = x^4 + x^3 + x^2 + x + 1$
 - (b) $p(x) = x^8 + x^4 + x^3 + x^2 + 1$
 - (c) $p(x) = x^{12} + x^6 + x^4 + x + 1$
5. Construct GF(128) as a vector space over GF(2).
6. When the 64 elements in GF(2^6) are arranged in conjugacy classes and their associated minimal polynomials. Find the minimal polynomial of the conjugacy class in which α^7 is an element.

Chapter 4

Linear Block Codes



This chapter deals with linear block codes covering their fundamental concepts, generator and parity check matrices, error-correcting capabilities, encoding and decoding, and performance analysis. The linear block codes discussed in this chapter are Hamming codes, cyclic codes, binary BCH codes, and Reed–Solomon codes.

4.1 Block Codes

The data stream is broken into blocks of k bits and each k -bit block is encoded into a block of n bits with $n > k$ bits as illustrated in Fig. 4.1. The n -bit block of the channel block encoder is called the code word. The code word is formed by adding $(n - k)$ parity check bits derived from the k message bits.

Some important properties of block codes are defined as

Block code rate

The block code rate (R) is defined as the ratio of k message bits and length of the code word n .

$$R = k/n \quad (4.1)$$

Code word weight

The weight of a code word or error pattern is the number of nonzero bits in the code word or error pattern. For example, the weight of a code word $c = (1, 0, 0, 1, 1, 0, 1, 0)$ is 4.

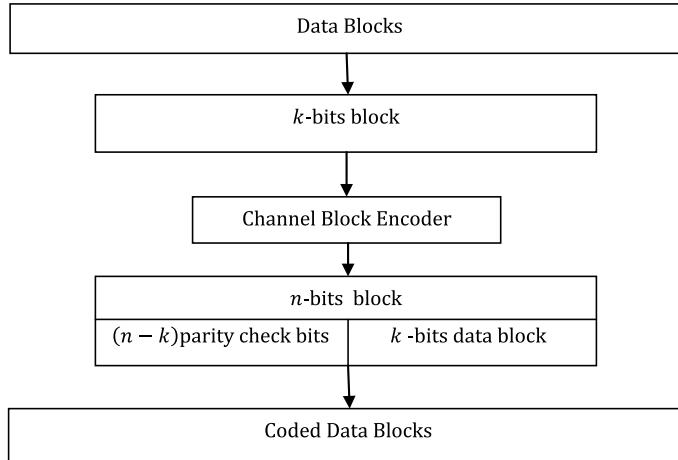


Fig. 4.1 Coded data stream

Hamming distance

The Hamming distance between two blocks v and w is the number of coordinates in which the two blocks differ.

$$d_{\text{Hamming}}(v, w) = d(v, w) = |\{i | v_i \neq w_i, i = 0, 1, \dots, n - 1\}| \quad (4.2)$$

Example 4.1 Consider the code words $v = (00100)$ and $w = (10010)$, then the Hamming distance $d_{\text{Hamming}}(v, w) = 3$.

Hamming distance allows for a useful characterization of the error detection and error correction capabilities of a block code as a function of the code's minimum distance.

The Minimum Distance of a Block Code

The minimum distance of a block code C is the minimum Hamming distance between all distinct pairs of code words in C .

A code with minimum distance d_{\min} can thus detect all error patterns of weight less than or equal to $(d_{\min} - 1)$.

A code with minimum distance d_{\min} can correct all error patterns of weight less than or equal to $[(d_{\min} - 1)/2]$.

Example 4.2 Consider the binary code C composed of the following four code words.

$$C = \{(00100), (10010), (01001), (11111)\}$$

Hamming distance of (00100) and $(10010) = 3$

Hamming distance of (10010) and $(01001) = 4$

Hamming distance of (00100) and $(01001) = 3$

Hamming distance of (10010) and (11111) = 3

Hamming distance of (00100) and (11111) = 4

Hamming distance of (01001) and (11111) = 3

Therefore, the minimum distance $d_{\min} = 3$.

4.2 Linear Block Codes

A block code C consisting of n -tuples $\{(c_0, c_1, \dots, c_{n-1})\}$ of symbols from GF(2) is said to be binary linear block code if and only if C forms a vector subspace over GF(2).

The code word is said to be systematic linear code word if each of the 2^k code words is represented as linear combination of k linearly independent code words.

4.2.1 Linear Block Code Properties

The two important properties of linear block codes are

- Property 1 The linear combination of any set of code words is a code word.
- Property 2 The minimum distance of a linear block code is equal to the minimum weight of any nonzero word in the code.

The two well-known bounds on the minimum distance are

1. Singleton Bound

The minimum distance of a (n, k) linear block code is bounded by

$$d_{\min} \leq n - k + 1 \quad (4.3a)$$

2. Hamming Bound

An (n, k) block code can correct up to t_{ec} errors per code word, provided that n and k satisfy the Hamming bound.

$$2^{n-k} \geq \sum_{i=0}^{t_{ec}} \binom{n}{i} \quad (4.3b)$$

The relation is the upper bound on the d_{\min} and is known as the Hamming bound.

Where $\binom{n}{i} = \frac{n!}{(n-i)!i!}$; $t_{ec} = (d_{\min} - 1)/2$.

4.2.2 Generator and Parity Check Matrices

Let $\{g_0, g_1, \dots, g_{k-1}\}$ be a basis of code words for the (n, k) linear block code C and $m = (m_0, m_1, \dots, m_{k-1})$ the message to be encoded. It follows from Theorem 3.1 that the codeword $c = (c_0, c_1, \dots, c_{n-1})$ for the message is uniquely represented by the following linear combination of g_0, g_1, \dots, g_{k-1}

$$c = m_0 g_0 + \dots + m_{k-1} g_{k-1} \quad (4.4)$$

for every code word $c \in C$. Since every linear combination of the basis elements must also be a code word, there is a one-to-one mapping between the set of k -bit blocks $(a_0, a_1, \dots, a_{k-1})$ over $GF(2)$ and the code words in C . A matrix G is constructed by taking the vectors in the basis as its rows.

$$G = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} g_{0,0} & g_{0,1} & \cdots & g_{0,n-1} \\ g_{1,0} & g_{1,1} & \cdots & g_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-1} \end{bmatrix} \quad (4.5)$$

This matrix is a generator matrix for the code C . It can be used to directly encode k -bit blocks in the following manner.

$$mG = (m_0, m_1, \dots, m_{k-1}) \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix} = m_0 g_0 + m_1 g_1 + \dots + m_{k-1} g_{k-1} = c$$

The dual space of a linear block code C is the dual code of C and a basis $\{h_0, h_1, \dots, h_{n-k-1}\}$ can be found for dual code of C , and the following parity check matrix can be constructed.

$$H = \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{n-k-1} \end{bmatrix} = \begin{bmatrix} h_{0,0} & h_{0,1} & \cdots & h_{0,n-1} \\ h_{1,0} & h_{1,1} & \cdots & h_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ h_{n-k-1,0} & h_{n-k-1,1} & \cdots & h_{n-k-1,n-1} \end{bmatrix} \quad (4.6)$$

In a systematic linear block code, the last k bits of the codeword are the message bits, that is,

$$c_i = m_{i-(n-k)}, \quad i = n - k, \dots, n - 1 \quad (4.7)$$

The first $n - k$ bits in the codeword are check bits generated from the k message bits according to

$$\begin{aligned} c_0 &= p_{0,0}m_0 + p_{1,0}m_1 + \dots + p_{k-1,0}m_{k-1} \\ c_1 &= p_{0,1}m_0 + p_{1,1}m_1 + \dots + p_{k-1,1}m_{k-1} \\ &\vdots \\ c_{n-k-1} &= p_{0,n-k-1}m_0 + p_{1,n-k-1}m_1 + \dots + p_{k-1,n-k-1}m_{k-1} \end{aligned}$$

The above equations can be written in matrix form as

$$[c_0, c_1, \dots, c_{n-1}] = [m_0, m_1, \dots, m_{k-1}] \begin{bmatrix} p_{0,0} & p_{0,1} & \dots & p_{0,n-k-1} & 1000 & \dots & 0 \\ p_{1,0} & p_{1,1} & \dots & p_{1,n-k-1} & 0100 & \dots & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots & \dots & \vdots \\ p_{k-1,0} & p_{k-1,1} & \dots & p_{k-1,n-k-1} & 0000 & \dots & 1 \end{bmatrix}_{k \times n} \quad (4.8)$$

or

$$c = mG \quad (4.9)$$

where G is the matrix on the right-hand side of Eq. (4.8). The $k \times n$ matrix G is called the generator matrix of the code and it has the form

$$G = \left[P : I_k \right]_{k \times n} \quad (4.10)$$

The matrix I_k is the identity matrix of order k , and P is an arbitrary k by $n - k$ matrix. When P is specified, it defines the (n, k) block code completely. The parity check matrix H corresponding to the above generator matrix G can be obtained as

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 & p_{0,0} & p_{1,0} & \dots & p_{k-1,0} \\ 0 & 1 & 0 & 0 & \dots & 0 & p_{0,1} & p_{1,1} & \dots & p_{k-1,1} \\ \vdots & \vdots & \vdots & & & & & & \vdots & \\ 0 & 0 & 0 & 0 & \dots & 1 & p_{0,n-k-1} & p_{1,n-k-1} & \dots & p_{k-1,n-k-1} \end{bmatrix} \quad (4.11)$$

$$H = \left[I_{n-k} : P^T \right] \quad (4.12)$$

The Parity Check Theorem

The parity check theorem states that “for an (n, k) linear block code C with $(n - k) \times n$ parity check matrix H , a code word $c \in C$ is a valid code word if and only if $cH^T = 0$.”

Example 4.3 Consider the following generator matrix of (7,4) block code. Find the code vector for the message vector $m = (1110)$ and check the validity of code vector generated.

$$G = \left[\begin{array}{ccc|cccc} 1 & 1 & 0 & | & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & | & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & | & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & | & 0 & 0 & 0 & 1 \end{array} \right]$$

Solution The code vector for the message block $m = (1110)$ is given by

$$c = mG = (1110) \left[\begin{array}{cccc|ccccc} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array} \right] = (0101110)$$

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

$$cH^T = [0101110] \left[\begin{array}{cccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{array} \right] = [000]$$

Hence, the generated code vector is valid.

4.2.3 Weight Distribution of Linear Block Codes

An (n, k) code contains 2^k code words with the Hamming weights between 0 and n . For $0 \leq j \leq n$, let W_j be the number of code words in C with Hamming weight j . The w_0, w_1, \dots, w_{n-1} are the weight distribution of C so that $w_0 + w_1 + w_2 + \dots + w_n = 2^k$. The weight distribution can be written as the polynomial $W(x) = w_0 + w_1x + w_2x^2 + \dots + w_{n-1}x^{n-1}$ which is called as weight enumerator. The weight distribution of a linear block code is related to the parity check matrix H by the following theorem.

“The minimum weight (or minimum distance) of an (n, k) linear block code with a parity check matrix H is equal to the minimum number of non zero columns in H whose vector sum is a zero vector.”

4.2.4 Hamming Codes

Hamming code is a linear block code capable of correcting single errors having a minimum distance $d_{\min} = 3$. It is very easy to construct Hamming codes. The parity check matrix H must be chosen so that no row in H^T is zero and the first $(n - k)$ rows of H^T form an identity matrix and all the rows are distinct.

We can select $2^{n-k} - 1$ distinct rows of H^T . Since the matrix H^T has n rows, for all of them to be distinct, the following inequality should be satisfied

$$2^{n-k} - 1 \geq n \quad (4.13)$$

implying that

$$(n - k) \geq \log_2(n + 1)$$

$$n \geq k + \log_2(n + 1) \quad (4.14)$$

Hence, the minimum size n for the code words can be determined from Eq. (4.14)

Example 4.4 Design a Hamming code with message block size of eleven bits.

Solution It follows from Eq. (4.14) that

$$n \geq 11 + \log_2(n + 1)$$

The smallest n that satisfies the above inequality is 15, and hence, we need a $(15, 11)$ block code. Thus, the transpose of the parity check matrix H will be 4 by 15 matrix. The first four rows of H^T will be 4×4 identity matrix. The last eleven rows are arbitrarily chosen, with the restrictions that no row is zero, and all the rows are distinct.

$$H = \left[\begin{array}{ccccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{array} \right]$$

$$H^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \dots & \dots & \dots & \dots \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} I_{n-k} \\ \dots \\ P^T \end{bmatrix}$$

Then, the generator matrix G is

$$G = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Example 4.5 Construct parity check and generator matrices for a (7, 4) Hamming code.

Solution The parity check matrix (H) and generator matrix (G) for a (7, 4) Hamming code are

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

4.2.5 Syndrome Table Decoding

Consider a valid code word c for transmission and let e be an error pattern introduced by the channel during transmission. Then, the received vector r can be written as

$$r = c + e \quad (4.15a)$$

Multiplying the r by the transpose of the parity check matrix gives the syndrome S which can be expressed as

$$\begin{aligned} S &= r H^T \\ &= (c + e) H^T \\ &= c H^T + e H^T \\ &= 0 + e H^T \\ &= e H^T \end{aligned} \quad (4.15b)$$

Thus, the syndrome vector is independent of the transmitted code word c and is only a function of the error pattern e . Decoding is performed by computing the syndrome of a received vector, looking up the corresponding error pattern, and subtracting the error pattern from the received word.

Example 4.6 Construct a syndrome decoding table for a $(7, 4)$ Hamming code.

Solution For a $(7, 4)$ Hamming code, there are $2^{(7-4)}$ error patterns (e) as follows

0	0	0	0	0	0	0
1	0	0	0	0	0	0
0	1	0	0	0	0	0
0	0	1	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	1	0	0
0	0	0	0	0	1	0
0	0	0	0	0	0	1

Table 4.1 Syndrome decoding table for a (7, 4) Hamming code

Error pattern	Syndrome
0 0 0 0 0 0 0	0 0 0
1 0 0 0 0 0 0	1 0 0
0 1 0 0 0 0 0	0 1 0
0 0 1 0 0 0 0	0 0 1
0 0 0 1 0 0 0	1 1 0
0 0 0 0 1 0 0	0 1 1
0 0 0 0 0 1 0	1 1 1
0 0 0 0 0 0 1	1 0 1

The syndrome for (7, 4) Hamming code is computed using the parity check matrix H as given in solution of Example 4.4 as follows

$$s = e * H^T$$

Thus, the syndrome decoding table for a (7, 4) Hamming code is as follows (Table 4.1).

4.2.6 Hamming Codes Decoding

Syndrome table is used to decode the Hamming codes. The syndrome table gives the syndrome value based on the simple relationship with parity check matrix. The single-error-correcting codes, i.e., Hamming codes are decoded by using syndrome value. Consider a code word c corrupted by e , an error pattern with a single one in the j th coordinate position results a received vector r . Let $\{h_0, h_1, \dots, h_{n-1}\}$ be the set of columns of the parity check matrix H . When the syndrome is computed, we obtain the transposition of the j th column of H .

$$s = r H^T = e H^T = (0, \dots, 0, 1, 0, \dots, 0) \begin{bmatrix} h_0^T \\ h_1^T \\ \vdots \\ h_{n-1}^T \end{bmatrix} = h_j^T \quad (4.16)$$

The above-mentioned process can be implemented using the following algorithm.

1. Compute the syndrome s for the received word. If $s = 0$, the received code word is the correct code word.
2. Find the position j of the column of H that is the transposition of the syndrome.
3. Complement the j th bit in the received codeword to obtain the corrected code word.

Example 4.7 Decode the received vector $r = (010000000000000)$ using the (15, 11) parity check matrix.

Solution

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

The received vector is $r = (010000000000000)$

The corresponding syndrome $s = r * H^T$ is

$$s = (0100)$$

The syndrome is the transposition of 1st column of H . Inverting the 1st coordinate of r , the following code word is obtained

$$c = (000000000000000)$$

Example 4.8 Decode the received vector $r = (001100011100000)$ vector using the (15, 11) parity check matrix vector.

Solution

The received vector is $r = (001100011100000)$

The corresponding syndrome $s = r * H^T$ is $s = (0011)$

The syndrome is the transposition of 7th column of H . Inverting the 7th coordinate of r , the following code word is obtained

$$c = (001100001100000)$$

4.3 Cyclic Codes

An (n, k) linear block code C is said to be a cyclic code if for every codeword $c = (c_0, c_1, \dots, c_{n-2}, c_{n-1}) \in C$, there is also a codeword $c_1 = (c_{n-1}, c_0, c_1, \dots, c_{n-2})$ obtained by shifting c cyclically one place to the right is also code word in C .

4.3.1 The Basic Properties of Cyclic Codes

Property 1 In an (n, k) cyclic code, there exists a unique polynomial called generator polynomial $g(x)$ of minimal degree $(n - k)$ of the following form:

$$g(x) = g_1x + g_2x^2 + \dots + g_{n-k-1}x^{n-k-1} + g_{n-k}x^{n-k} \quad (4.17)$$

Property 2 Every code polynomial in an (n, k) cyclic code is multiple of $g(x)$. Thus, it can be expressed as $c(x) = m(x)g(x)$ where $m(x)$ is a polynomial over GF(2) of degree $k - 1$ or less.

Property 3 The generator polynomial $g(x)$ of an (n, k) cyclic code over GF(2) divides $x^n + 1$.

Property 4 The generator polynomial $g(x)$ and the parity check matrix $h(x)$ are factor of the polynomial $1 + x^n$.

In modulo-2 arithmetic $1 + x^n$ has the same value $1 - x^n$.

Example 4.9 Let C_1 be the binary cyclic code of length 15 generated by $g(x) = x^5 + x^4 + x^2 + 1$. Compute the code polynomial in C_1 and the associated code word for the message polynomial $m(x) = x^9 + x^4 + x^2 + 1$ using the polynomial multiplication encoding technique.

Solution

$$m(x) = x^9 + x^4 + x^2 + 1; g(x) = x^5 + x^4 + x^2 + 1$$

code polynomial

$$\begin{aligned} c(x) &= m(x)g(x) \\ &= x^{14} + x^{13} + x^{11} + x^8 + x^7 + x^5 + x^4 + 1 \end{aligned}$$

Code word = (100011011001011).

Example 4.10 Let C_1 be the binary cyclic code of length 15 generated by $g(x) = x^5 + x^4 + x^2 + 1$. Determine the dimensions of C_1 and compute the number of code words in C_1 .

Solution Since the order of the generator polynomial is 5. The C_1 has dimension $(15, 10)$ with $k = (15 - 5) = 10$ and contains 2^{15-5} code words.

Example 4.11 Let C_1 be the binary cyclic code of length 15 generated by $g(x) = x^5 + x^4 + x^2 + 1$. Compute the parity check polynomial for C_1 and show that $g(x)$ is a valid generator polynomial.

Solution

$$g(x) = x^5 + x^4 + x^2 + 1$$

The parity check polynomial for C_1 is

$$h(x) = \frac{x^{15} + 1}{g(x)} = x^{10} + x^9 + x^8 + x^6 + x^5 + x^2 + 1$$

$g(x)$ is valid generator polynomial since it has the minimum polynomials $x^4 + x + 1$ and $x + 1$ as factors, i.e., $g(x) = (x^4 + x + 1)(x + 1)$.

4.3.2 Encoding Algorithm for an (n, k) Cyclic Codes

In an (n, k) cyclic code C with generator polynomial $g(x)$, let $m = (m_0, m_1, \dots, m_{k-1})$ is the message block. By multiplying the message polynomial $m(x)$ by x^{n-k} , we obtain a polynomial $x^{n-k}m(x) = m_0x^{n-k} + m_1x^{n-k+1} + \dots + m_{k-1}x^{n-1}$ of degree $n - 1$ or less. Now, dividing $x^{n-k}m(x)$ by $g(x)$ yields

$$x^{n-k}m(x) = q(x)g(x) + p(x) \quad (4.18)$$

where $q(x)$ and $p(x)$ are the quotient and remainder, respectively.

Equation (4.18) can be rearranged as

$$p(x) + x^{n-k}m(x) = q(x)g(x) \quad (4.19)$$

Equation (4.19) shows that $p(x) + x^{n-k}m(x)$ is divisible by $g(x)$. Hence, it must be a valid code polynomial $c(x) = p(x) + x^{n-k}m(x)$ of the (n, k) cyclic code C with generator polynomial $g(x)$. The n -tuple representation of the code polynomial $c(x)$ is

$$c = (p_0, p_1, \dots, p_{n-k-1}, m_0, m_1, \dots, m_{k-1}) \quad (4.20)$$

The systematic encoding algorithm is summarized as

- Step 1 Multiply the message polynomial $m(x)$ by x^{n-k} .
- Step 2 Divide the result of step 1 by the generator polynomial $g(x)$. Let $d(x)$ be the remainder.
- Step 3 Set $c(x) = x^{n-k}m(x) - d(x)$.

Example 4.12 Let C_1 be the binary cyclic code of length 15 generated by $g(x) = x^5 + x^4 + x^2 + 1$. Compute the code polynomial in C_1 and the associated code word for the message polynomial $m(x) = x^8 + x^7 + x^6 + x^5 + x^4$ using the systematic encoding technique. Verify that the message has been systematically encoded.

Solution

$$g(x) = x^5 + x^4 + x^2 + 1; m(x) = x^8 + x^7 + x^6 + x^5 + x^4$$

$$\text{Step 1: } x^5m(x) = x^5(x^8 + x^7 + x^6 + x^5 + x^4) = x^{13} + x^{12} + x^{11} + x^{10} + x^9$$

$$\frac{x^8 + x^6 + x^5 + x^2 + 1}{x^5 + x^4 + x^2 + 1}$$

$$\begin{array}{r} x^{13} + x^{12} + x^{11} + x^{10} + x^9 \\ x^{13} + x^{12} + x^{10} + x^8 \\ \hline x^{11} + x^9 + x^8 \\ x^{11} + x^{10} + x^8 + x^6 \\ \hline \end{array}$$

Step 2:

$$\begin{array}{r} x^{10} + x^9 + x^6 \\ x^{10} + x^9 + x^7 + x^5 \\ \hline x^7 + x^6 + x^5 \\ x^7 + x^6 + x^4 + x^2 \\ \hline x^5 + x^4 + x^2 \\ x^5 + x^4 + x^2 + 1 \\ \hline \end{array}$$

$$1 = d(x)$$

$$\text{Step 3: } c_m(x) = x^{13} + x^{12} + x^{11} + x^{10} + x^9 + 1 \leftrightarrow c_m = (100000000111110).$$

Example 4.13 Construct parity check and generator matrices for binary cyclic code of length 15 generated by $g(x) = x^5 + x^4 + x^2 + 1$.

Solution The systematic generator matrix is obtained by selecting as rows those code words associated with the message blocks (1000000000), (0100000000), (0010000000), (0001000000), (0000100000), (0000010000), (0000001000), (0000000100), (0000000010), and (1000000001).

$m(x)$	Code polynomial $c(x)$	Codeword
1	$1 + x^2 + x^4 + x^5$	$\leftrightarrow (101011000000000)$
x	$1 + x + x^2 + x^3 + x^4 + x^6$	$\leftrightarrow (1111010000000)$
x^2	$1 + x + x^3 + x^7$	$\leftrightarrow (11010001000000)$
x^3	$x + x^2 + x^4 + x^8$	$\leftrightarrow (011010001000000)$
x^4	$1 + x^3 + x^4 + x^9$	$\leftrightarrow (100110000100000)$
x^5	$1 + x + x^2 + x^{10}$	$\leftrightarrow (11100000010000)$
x^6	$x + x^2 + x^3 + x^{11}$	$\leftrightarrow (01110000001000)$
x^7	$x^2 + x^3 + x^4 + x^{12}$	$\leftrightarrow (001110000000100)$
x^8	$1 + x^2 + x^3 + x^{13}$	$\leftrightarrow (101100000000010)$
x^9	$x + x^3 + x^4 + x^{14}$	$\leftrightarrow (010110000000001)$

The generator matrix (G) and parity check matrix (H) for the cyclic code are

$$G = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The corresponding parity check matrix is,

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

4.3.3 Encoder for Cyclic Codes Using Shift Registers

The systematic encoder for cyclic codes is shown in Fig. 4.2. The rectangular boxes represent flip flops which reside either in 0 or 1 state. The encoder operation is as follows.

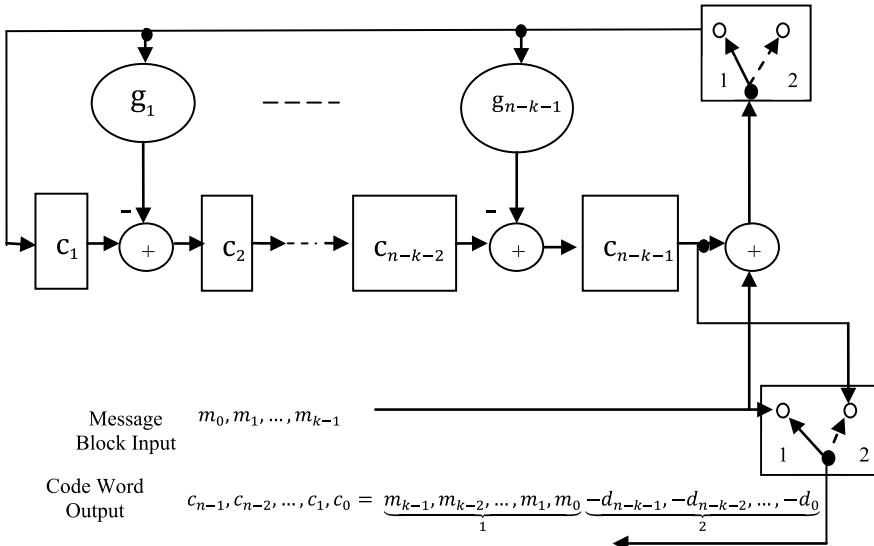


Fig. 4.2 Encoding circuit for (n, k) cyclic code

1. The switches are placed in position in 1. The k message bits are sent to the modulator and placed at the end of the systematic code word. As soon as the k th message bit fed into the shift register, the flip flops of the shift register contain $(n - k)$ parity bits.
2. The switches are moved to the position 2 to break the feedback connection.
3. The parity bits in the shift register are shifted out into the transmitter to form the parity bits of the systematic codeword.

Example 4.14 Construct the shift register encoder for a cyclic code of length 7 generated by $g(x) = x^4 + x^3 + x^2 + 1$ and obtain the code word for message $m = (010)$.

Solution The shift register for encoding the $(7, 3)$ cyclic code with generator polynomial $g(x) = x^4 + x^3 + x^2 + 1$ is shown in Fig. 4.3. The given message bits are 010. The contents of the shift register are shown in Table 4.2. Hence, the four parity check bits are 0111. Therefore, the codeword output is 0111010.

4.3.4 Shift Register Encoders for Cyclic Codes

Suppose the code word $(c_0, c_1, \dots, c_{n-1})$ is transmitted over a noisy channel resulting in the received word $(r_0, r_1, \dots, r_{n-1})$. Let the received word be represented by a polynomial of degree $n - 1$ or less as

$$r(x) = r_0 + r_1x + \dots + r_{n-1}x^{n-1} \quad (4.21)$$

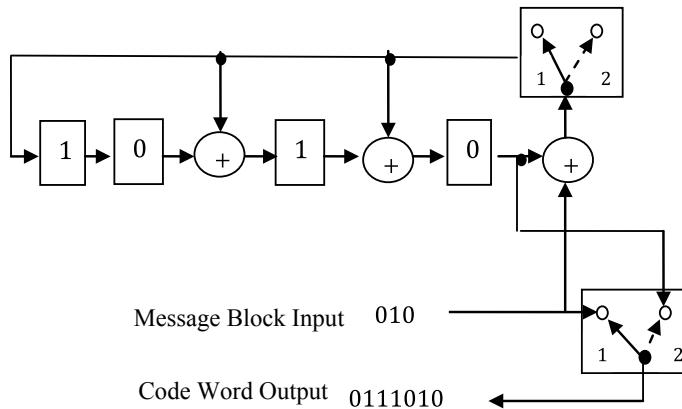


Fig. 4.3 Encoder for an (7,3) cyclic code generated by $g(x) = x^4 + x^3 + x^2 + 1$

Table 4.2 Contents of the shift register in the encoder of Fig. 4.3 for message sequence (010)

Shift	Input	Register codewords			
		0	0	0	0
1	0	0	0	0	0
2	1	1	0	1	0
3	0	0	1	1	1

Dividing the $r(x)$ by $g(x)$ results in the following

$$r(x) = q(x)g(x) + s(x) \quad (4.22)$$

where $q(x)$ is the quotient and $s(x)$ is the remainder known as syndrome. The $s(x)$ is a polynomial of degree $n - k - 1$ or less, and its coefficients make up the $(n - 1)$ -by-1 syndrome s . An error in the received word is detected only when the syndrome polynomial $s(x)$ is nonzero.

Syndrome Calculator

The syndrome calculator shown in Fig. 4.4 is similar to the encoder shown in Fig. 4.2. The only difference is that the received bits are fed from left into the $(n - k)$ stages of the feedback shift register. At the end of last received bit shifting, the contents of the shift register contain the desired syndrome s . If the syndrome is zero, there are no transmission errors in the received word or else the received code word contains transmission error. By knowing the value of syndrome, we can determine the corresponding error pattern and also make the appropriate correction.

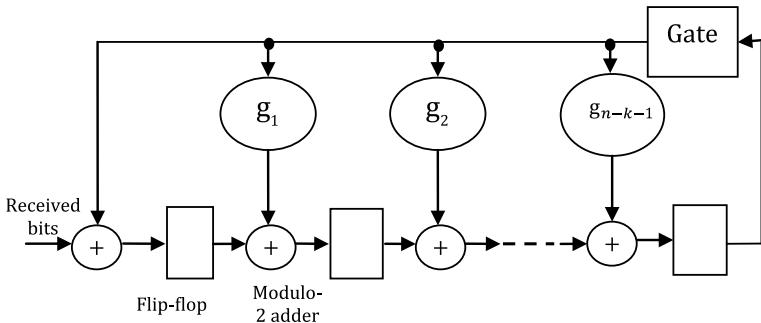


Fig. 4.4 Syndrome calculator

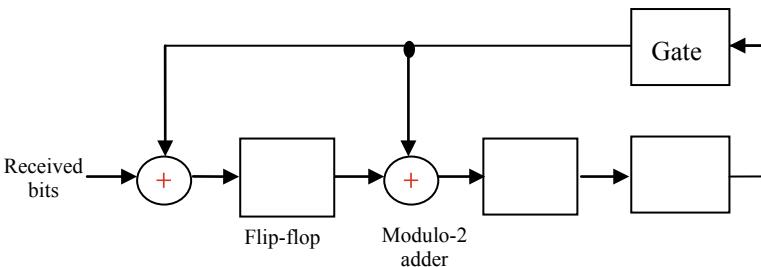


Fig. 4.5 Syndrome calculator of Example 4.15

Example 4.15 Consider the (7, 4) Hamming code generator polynomial $g(x) = x^3 + x + 1$ and the transmitted code word 1100101. Show the fifth bit of the received word is an error.

Solution

Given $g(x) = x^3 + x + 1$

Transmitted codeword = 1100101

By considering the fifth bit as an error, the received word = 1110101 (Fig. 4.5).

At the end of the seventh shift, the contents of the shift register (syndrome) are 001. The nonzero value of the syndrome indicates the error, and the error pattern for the syndrome 001 is 0010000 from Table 4.1. This shows that the fifth bit of the received word is an error (Table 4.3).

4.3.5 Cyclic Redundancy Check Codes

Cyclic redundancy check (CRC) is a method of detecting accidental changes/burst errors in communication channel. CRC codes are implemented from cyclic codes

Table 4.3 Contents of the shift register in the encoder of Fig. 4.6

Shift	Input bit	Contents of shift register
		000
1	1	100
2	0	010
3	1	101
4	0	100
5	1	110
6	1	111
7	1	001

and hence the name, even they are generally not cyclic. The CRC generation and checking are shown in Fig. 4.6, in which a group of error control bits (which is the remainder of a polynomial division of a message polynomial by a generator polynomial) is appended to the end of the message block.

The encoding of an (n, k) CRC code is same as that of cyclic code generation. This is illustrated through the following example.

Example 4.16 Let C_1 be the binary CRC code of length 12 generated by $g(x) = x^4 + x^3 + 1$. Compute the code polynomial in C_1 and the associated code word for the message polynomial $m(x) = x^7 + x^6 + x^5 + x^2 + x$ using the systematic encoding technique.

Solution $g(x) = x^4 + x^3 + 1$; $m(x) = x^7 + x^6 + x^5 + x^2 + x$;

$$\text{Step 1: } x^4m(x) = x^4(x^7 + x^6 + x^5 + x^2 + x) = x^{11} + x^{10} + x^9 + x^6 + x^5$$

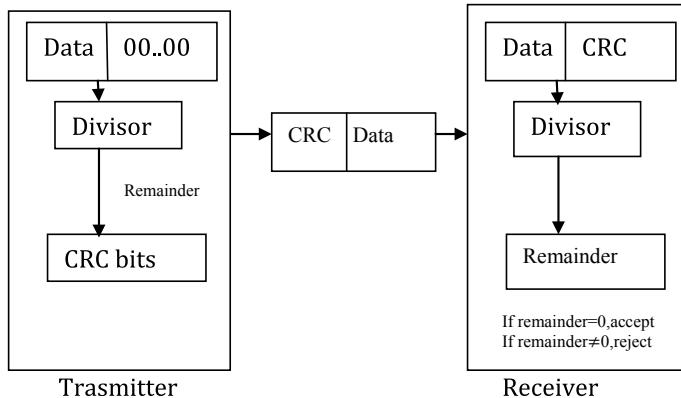


Fig. 4.6 CRC generation and checking

Step 2:

$$\begin{array}{r}
 & x^7 + x^5 + x^4 + x^2 + x \\
 \hline
 x^4 + x^3 + 1) & x^{11} + x^{10} + x^9 + x^6 + x^5 \\
 & x^{11} + x^{10} + x^7 \\
 \hline
 & x^9 + x^7 + x^6 + x^5 \\
 & x^9 + x^8 + x^5 \\
 \hline
 & x^8 + x^7 + x^6 \\
 & x^8 + x^7 + x^4 \\
 \hline
 & x^6 + x^4 \\
 & x^6 + x^5 + x^2 \\
 \hline
 & x^5 + x^4 + x^2 \\
 & x^5 + x^4 + x \\
 \hline
 & x^2 + x = d(x)
 \end{array}$$

Therefore, the appended bits are 0110

Step 3: $c(x) = x^5m(x) + d(x) = x^{11} + x^{10} + x^9 + x^6 + x^5 + x^2 + x \leftrightarrow c = (011001100111)$.

The following three CRC codes are given in Table 4.4 that have become international standard.

4.4 BCH Codes

BCH codes are a subclass of cyclic codes. The BCH codes are introduced independently by Bose, Ray-Chaudhuri, and Hocquenghem. For $m > 3$ and $t_{ec} < 2^{m-1}$, there exists a BCH code with parity check bits $(n - k) \leq mt_{ec}$ and $d_{min} \geq 2t_{ec} + 1$.

4.4.1 BCH Code Design

If a primitive element α of $GF(2^m)$ is chosen, then the generator polynomial $g(x)$ of the t_{ec} -error-correcting binary BCH code of length $2^m - 1$ is the minimum degree polynomial over $GF(2)$ having $\alpha, \alpha^2, \alpha^3, \dots, \alpha^{2t_{ec}}$ as roots. Suppose $\phi_i(\alpha)$ be the minimal polynomial of α^i , $1 \leq i \leq 2t_{ec}$ for t_{ec} -error-correcting binary BCH code. Then the generator polynomial $g(x)$ is the least common multiple (LCM) of $\phi_1(\alpha), \phi_2(\alpha), \dots, \phi_{2t_{ec}}(\alpha)$, i.e.,

Table 4.4 International standard CRC codes

CRC code	Description	Error detection capability	Burst error detection capability
CRC-12 2047	$g(x) = x^{12} + x^{11} + x^3 + x^2 + x + 1$ $= (x^{11} + x^2 + 1)(x + 1)$ Code length: Number of parity bits: 12 $d_{\min} = 4$	All one bit errors, two, and three bit errors of length up to 32767. All error pattern with an odd number of error if the generator polynomial $g(x)$ for the code has an even number of nonzero coefficients	All burst errors up to length 16
CRC-16 32,767	$g(x) = x^{16} + x^{15} + x^2 + 1$ $= (x^{15} + x + 1)(x + 1)$ Code length: 32,767 Number of parity bits: 16 $d_{\min} = 4$	All one bit errors, two and three bit errors of length up to 32767. All error pattern with an odd number of error if the generator polynomial $g(x)$ for the code has an even number of nonzero coefficients	All burst errors up to length 16
CRC-CCITT 32,767	$g(x) = x^{16} + x^{12} + x^5 + 1$ $= (x^{15} + x^{14} + x^{13} + x^{12}$ $+ x^4 + x^3 + x^2 + x + 1)(x + 1)$ Code length: 32,767 Number of parity bits: 16 $d_{\min} = 4$	All one bit errors, two and three bit errors of length up to 32767. All error pattern with an odd number of error if the generator polynomial $g(x)$ for the code has an even number of nonzero coefficients	All burst errors up to length 16

$$g(x) = \text{LCM}\{\phi_1(\alpha), \phi_2(\alpha), \dots, \phi_{2t_{ec}}(\alpha)\} \quad (4.23)$$

Let $c(x) = c_0 + c_1\alpha^i + c_2\alpha^{2i} + \dots + c_{2^m-2}\alpha^{(2^m-2)i}$ be a code polynomial. Since $c(x)$ is divisible by $g(x)$, a root of $g(x)$ is also a root of $c(x)$. Hence, for $1 \leq i \leq 2t_{ec}$,

$$c(\alpha^i) = c_0 + c_1\alpha^i + c_2\alpha^{2i} + \dots + c_{2^m-2}\alpha^{(2^m-2)i} = 0 \quad (4.24)$$

Equation (4.24) can be rewritten in matrix form as:

$$(c_0, c_1, \dots, c_{2^m-2}) \begin{pmatrix} 1 \\ \alpha^i \\ \alpha^{2i} \\ \vdots \\ \alpha^{(2^m-2)i} \end{pmatrix} = 0 \quad (4.25)$$

It follows that $c.H^T = 0$ for every codeword $c = (c_0, c_1, \dots, c_{2^m-2})$ in the t_{ec} -error-correcting BCH code of length $2^m - 1$ generated by $g(x)$. Hence, for $g(x)$ the corresponding $2t \times (2^m - 1)$ matrix over $\text{GF}(2^m)$ can be formed as:

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \dots & \alpha^{(2^m-2)} \\ 1 & \alpha^2 & (\alpha^2)^2 & \dots & (\alpha^2)^{2^m-2} \\ 1 & \alpha^3 & (\alpha^3)^2 & \dots & (\alpha^3)^{2^m-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{2t} & (\alpha^{2t})^2 & \dots & (\alpha^{2t})^{2^m-2} \end{bmatrix} \quad (4.26)$$

If $\phi(x)$ is the minimal polynomial of an element β of the $\text{GF}(2^m)$ and l ($l \leq m$) is the smallest positive integer such that $\beta^{2^l} = \beta$, then from Theorem 3.7 $\phi(x)$ of degree m or less is given by

$$\phi(x) = \prod_{i=0}^{l-1} (X - \beta^{2^i}) \quad (4.27)$$

The conjugates of β are the roots of $\Phi_\beta(x)$ of the form β^{2^i} , $1 < i < l - 1$

From Theorem 3.6, the roots of $\Phi_\beta(x)$ having the conjugacy class will have the same minimal polynomial.

The stepwise procedure to find the minimal polynomial $\Phi_\beta(x)$ is as follows:

Step 1. Determine the conjugates class of β

Step 2. Obtain $\Phi_\beta(x)$ using Eq. (4.27)

The design procedure of t_{ec} -error-correcting binary BCH code of length n is as follows:

1. Choose a primitive root α in a field $\text{GF}(2^m)$.

2. Select $2t_{ec}$ consecutive powers of α .
3. Obtain the minimal polynomials for all the $2t_{ec}$ consecutive powers of α having the same minimal polynomial for the roots in the same conjugacy class.
4. Obtain the generator polynomial $g(x)$ by taking the least common multiple of the minimal polynomials for the $2t_{ec}$ consecutive powers of α .

The construction of BCH codes are illustrated through the following examples.

Example 4.17 Compute a generator polynomial for a binary BCH code of length 15 and minimum distance 3, and find the code rate.

Solution Since 15 is of the form $2^m - 1$, the BCH codes are primitive. Let α be a primitive element in the field GF(16) generated by the primitive polynomial $1+x+x^4$. The elements of the field GF(16) are given in Table 3.3. Since the code is to be single-error-correcting (minimum distance = 3), the generator polynomial thus must have α and α^2 as roots. The α and α^2 are conjugate elements and have the same minimal polynomial, which is $1 + x + x^4$. The generator polynomial is thus

$$g(x) = 1 + x + x^4$$

Since the degree of $g(x)$ is 4, the BCH code generator by $g(x)$ is a (15,11) code. The rate of the code is

$$R = \frac{k}{n} = \frac{11}{15}$$

Example 4.18 Design a double-error-correcting binary BCH code of length 15.

Solution Since 15 is of the form $2^m - 1$, the BCH codes are primitive. Let α be a primitive element in the field GF(16) generated by the primitive polynomial $1+x+x^4$. The elements of the field GF(16) are given in Table 3.3.

Since the code is to be double-error-correcting, the generator polynomial thus must have $\alpha, \alpha^2, \alpha^3, \alpha^4$ as roots.

The α, α^2 and α^4 are conjugates and have the same minimal polynomial, which is $1 + x + x^4$. Thus,

$$\phi_\alpha(x) = \phi_{\alpha^2}(x) = \phi_{\alpha^4}(x) = 1 + x + x^4$$

By letting $\beta = \alpha^3$

$$\beta^{2^4} = (\alpha^3)^{16} = \alpha^{48} = \alpha^{45}\alpha^3 = 1.\alpha^3 = \alpha^3$$

Therefore $l = 4$, and from Eq. (4.28), the minimal polynomial $\phi_{\alpha^3}(x)$ is given by

$$\begin{aligned}
\phi_{\alpha^3}(x) &= \prod_{i=0}^{l-1} (x - \beta^{2^i}) = \prod_{i=0}^{4-1} (x - \beta^{2^i}) \\
&= (x - \alpha^3)(x - \alpha^6)(x - \alpha^{12})(x - \alpha^{24}) \\
\phi_{\alpha^3}(x) &= (x - \alpha^3)(x - \alpha^6)(x - \alpha^{12})(x - \alpha^9) \\
&= 1 + x + x^2 + x^3 + x^4
\end{aligned}$$

Hence,

$$\begin{aligned}
g(x) &= (1 + x + x^4)(1 + x + x^2 + x^3 + x^4) \\
&= 1 + x^4 + x^6 + x^7 + x^8
\end{aligned}$$

Since the degree of $g(x)$ is 8, the BCH code generator by $g(x)$ is a (15,7) code with minimum distance 5.

Example 4.19 Design a triple-error-correcting binary BCH code of length 63.

Solution Let α be a primitive element in the field GF(16) generated by the primitive polynomial $1 + x + x^6$. The elements of the field GF(64) are given in Table 3.8. Since the code is to be triple-error-correcting, the generator polynomial thus must have $\alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6$ as roots. α, α^2 and α^4 are conjugate elements and have the same minimal polynomial, which is $1 + x + x^6$.

Thus,

$$\phi_\alpha(x) = \phi_{\alpha^2}(x) = \phi_{\alpha^4}(x) = 1 + x + x^6$$

The elements α^3 and α^6 are conjugates and have the same minimal polynomial. By letting $\beta = \alpha^3$

$$\beta^{2^6} = (\alpha^3)^{64} = \alpha^{192} = \alpha^{63}\alpha^{63}\alpha^{63}\alpha^3 = 1 \cdot \alpha^3 = \alpha^3$$

Therefore $l = 6$, and from Eq. (4.28), the minimal polynomials $\phi_{\alpha^3}(x)$ and $\phi_{\alpha^6}(x)$ are the same and given by

$$\begin{aligned}
\phi_{\alpha^3}(x) &= \phi_{\alpha^6}(x) = \prod_{i=0}^{l-1} (x - \beta^{2^i}) = \prod_{i=0}^{6-1} (x - \beta^{2^i}) \\
&= (x - \alpha^3)(x - \alpha^6)(x - \alpha^{12})(x - \alpha^{24})(x - \alpha^{48})(x - \alpha^{96}) \\
&= (x - \alpha^3)(x - \alpha^6)(x - \alpha^{12})(x - \alpha^{24})(x - \alpha^{48})(x - \alpha^{33}) \\
&= 1 + x + x^2 + x^4 + x^6
\end{aligned}$$

By letting $\beta = \alpha^5$

$$\beta^{2^6} = (\alpha^5)^{64} = \alpha^{320} = \alpha^{63}\alpha^{63}\alpha^{63}\alpha^{63}\alpha^3 = 1 \cdot \alpha^5 = \alpha^5$$

Therefore $l = 6$, and from Eq. (4.28), the minimal polynomial $\phi_{\alpha^5}(x)$ is given by

$$\begin{aligned}\phi_{\alpha^5}(x) &= \prod_{i=0}^{l-1} (x - \beta^{2^i}) = \prod_{i=0}^{6-1} (x - \beta^{2^i}) \\ &= (x - \alpha^{53is})(x - \alpha^{10})(x - \alpha^{20})(x - \alpha^{40})(x - \alpha^{80})(x - \alpha^{160}) \\ &= (x - \alpha^5)(x - \alpha^{10})(x - \alpha^{20})(x - \alpha^{40})(x - \alpha^{17})(x - \alpha^{34}) \\ &= 1 + x + x^2 + x^5 + x^6\end{aligned}$$

It follows from Eq. (4.24) that the generator polynomial of the triple-error-correcting BCH code of length 63 is given by

$$\begin{aligned}g(x) &= (1 + x + x^6)(1 + x + x^2 + x^4 + x^6)(1 + x + x^2 + x^5 + x^6) \\ &= 1 + x + x^2 + x^3 + x^6 + x^7 + x^9 + x^{15} + x^{16} + x^{17} + x^{18}\end{aligned}$$

Since the degree of $g(x)$ is 18, the BCH code generator by $g(x)$ is a (63, 45) code with minimum distance 7.

Example 4.20 Construct generator and parity check matrices for a single-error-correcting BCH code of length 15.

Solution A parity check matrix for this code is obtained by using Eq. (4.27) as

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \dots & \alpha^{13} & \alpha^{14} \\ 1 & \alpha^2 & \alpha^4 & \dots & \alpha^{11} & \alpha^{13} \end{bmatrix}$$

This parity check matrix has redundancy because α and α^2 conjugates. Hence, the parity check matrix without redundancy is

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \dots & \alpha^{13} & \alpha^{14} \end{bmatrix}$$

Note that the entries of H are elements in $GF(2^4)$. Each element in $GF(2^4)$ can be represented by a 4-tuples over $GF(2)$. If each entry of H is replaced by its corresponding 4-tuples over $GF(2)$ arranged in column form, we obtain a binary parity check matrix for the code as follows

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The corresponding generator matrix is

$$G = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

4.4.2 Berlekamp's Algorithm for Binary BCH Codes Decoding

Let code polynomial $c(x) = c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_1x + c_0$,
an error polynomial $e(x) = e_{n-1}x^{n-1} + e_{n-2}x^{n-2} + \cdots + e_1x + e_0$
and received polynomial $r(x) = r_{n-1}x^{n-1} + r_{n-2}x^{n-2} + \cdots + r_1x + r_0$.
Then, $r(x)$ can be written as

$$r(x) = c(x) + e(x) \quad (4.28)$$

Let $S = [S_1 S_2 \dots S_{2t_{ec}}]$ be syndrome sequence with $2t_{ec}$ known syndrome components. Then the syndrome polynomial can be written as

$$S(x) = S_{2t_{ec}}x^{2t_{ec}} + S_{2t_{ec}-1}x^{2t_{ec}-1} + \cdots + S_1x \quad (4.29)$$

where t_{ec} stands for error-correcting capability. By evaluating the received polynomial at $2t_{ec}$ zeros, the syndromes $S_1 S_2 \dots S_{2t_{ec}}$ can be obtained. Thus,

$$S_i = r(\alpha^i) = r_{n-1}(\alpha^i)^{n-1} + r_{n-2}(\alpha^i)^{n-2} + \cdots + r_1(\alpha^i) + r_0 \quad (4.30)$$

for $1 \leq i \leq 2t_{ec}$

The syndrome sequence $S_1 S_2 \dots S_{2t_{ec}}$ can be rewritten as

$$S_i = e(\alpha^i) = e_{n-1}(\alpha^i)^{n-1} + e_{n-2}(\alpha^i)^{n-2} + \cdots + e_1(\alpha^i) + e_0 \quad (4.31)$$

for $1 \leq i \leq 2t_{ec}$

Assuming that the received word r has v errors in positions j_1, j_2, \dots, j_v , the error locator polynomial can be expressed as

$$\begin{aligned}\Lambda(x) &= \Lambda_v x^v + \Lambda_{v-1} x^{v-1} + \cdots + \Lambda_1 x + 1 \\ &= (1 - \alpha^{j_1} x)(1 - \alpha^{j_2} x) \cdots (1 - \alpha^{j_v} x)\end{aligned}\quad (4.32)$$

The error magnitude polynomial is defined as

$$\Omega(x) = \Lambda(x)(1 + S(x)) \bmod x^{2t_{ec}+1} \quad (4.33)$$

This is useful in non-binary decoding.

Berlekamp's algorithm proceeds for binary decoding of BCH codes iteratively by breaking down into a series of smaller problems of the form

$$[1 + S(x)]\Lambda^{(2n)}(x) \equiv (1 + \Omega_2 x^2 + \Omega_4 x^4 + \cdots + \Omega_{2n} x^{2n}) \bmod x^{2n+1} \quad (4.34)$$

where n runs from 1 to t_{ec} . The flowchart of the Berlekamp's iterative algorithm is shown in Fig. 4.7.

4.4.3 Chien Search Algorithm

A Chien search is shown in Fig. 4.8. The Chien search is a systematic means of evaluating the error locator polynomial at all elements in a field $\text{GF}(2^m)$. Each coefficient of the error locator polynomial is repeatedly multiplied by α^i where α is primitive in $\text{GF}(2^m)$. Each set of products is then summed to obtain $A_i = \Lambda(\alpha^i) - 1$. If α^i is a root of $\Lambda(x)$, then $A_i = \Lambda(\alpha^i) - 1$ and an error is indicated at the coordinate associated with $\alpha^{-i} = \alpha^{n-i}$.

Example 4.21 Let the transmission code be the triple-error-correcting binary BCH code of length 15. The generator polynomial is $g(x) = 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10}$. Use Berlekamp's algorithm to decode the following received vector $r = (000101000000100)$.

Solution For double error correction, the generator polynomial $g(x) = 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10}$ has roots which include six consecutive powers of α : $(\alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6)$ where α is primitive in $\text{GF}(16)$.

The received vector is $r = (000101000000100) \leftrightarrow r(x) = x^3 + x^5 + x^{12}$

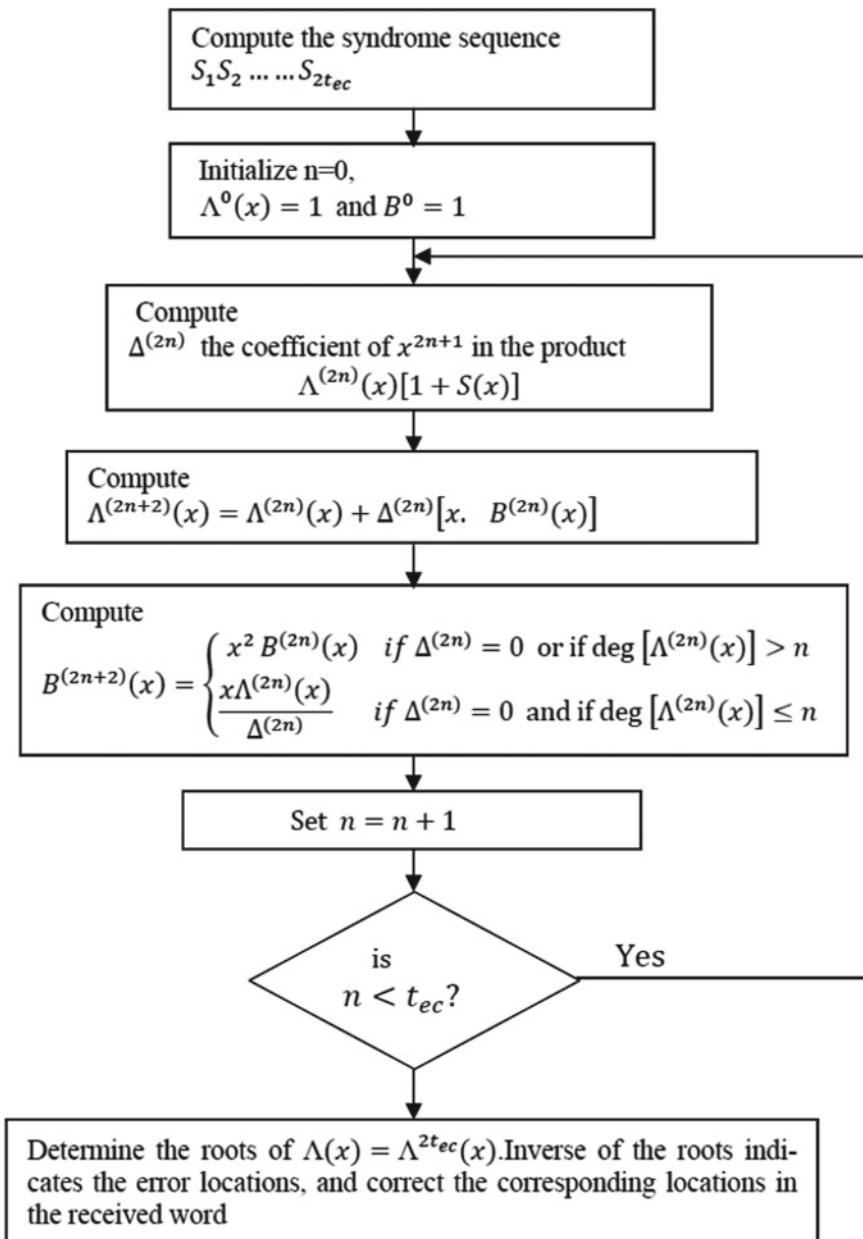


Fig. 4.7 Berlekamp iterative algorithm for decoding binary BCH codes

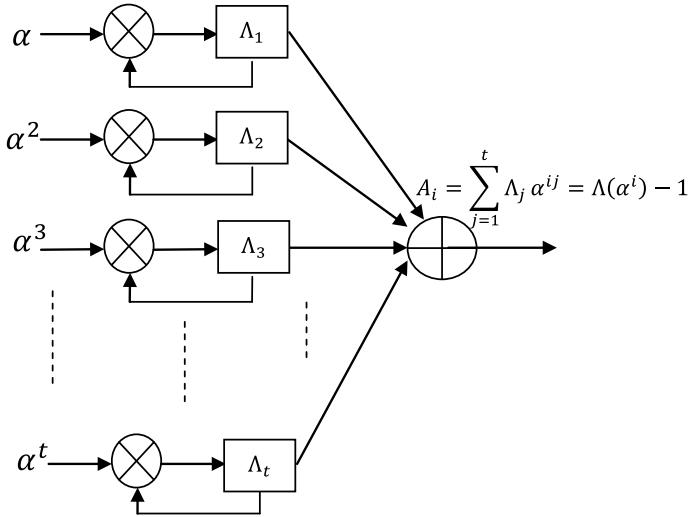


Fig. 4.8 Chien search circuit

The syndrome polynomial is written as

$$\begin{aligned}
 S(x) &= S_1x + S_2x^2 + S_3x^3 + S_4x^4 + S_5x^5 + S_6x^6 \\
 S_1 &= r(\alpha) = \alpha^3 + \alpha^5 + \alpha^{12} = 1 \\
 S_2 &= r(\alpha^2) = \alpha^6 + \alpha^{10} + \alpha^{24} = 1 \\
 S_3 &= r(\alpha^3) = \alpha^9 + \alpha^{15} + \alpha^{36} = \alpha^{10} \\
 S_4 &= r(\alpha^4) = \alpha^{12} + \alpha^{20} + \alpha^{48} = 1 \\
 S_5 &= r(\alpha^5) = \alpha^{15} + \alpha^{25} + \alpha^{60} = \alpha^{10} \\
 S_6 &= r(\alpha^6) = \alpha^{18} + \alpha^{30} + \alpha^{72} = \alpha^5
 \end{aligned}$$

Since $\alpha^{15} = 1$;

$$\begin{aligned}
 \alpha^{24} &= \alpha^{15}\alpha^9 = \alpha^9; \alpha^{36} = \alpha^{15}\alpha^{15}\alpha^6 = \alpha^6; \alpha^{20} = \alpha^{15}\alpha^5 \\
 &= \alpha^5; \alpha^{48} = \alpha^{15}\alpha^{15}\alpha^{15}\alpha^3 = \alpha^3
 \end{aligned}$$

$$\begin{aligned}
 \alpha^{25} &= \alpha^{15}\alpha^{10} = \alpha^{10}; \alpha^{60} = \alpha^{15}\alpha^{15}\alpha^{15}\alpha^{15} = 1; \alpha^{30} = \alpha^{15}\alpha^{15} = 1; \alpha^{72} \\
 &= \alpha^{15}\alpha^{15}\alpha^{15}\alpha^{15}\alpha^{12} = \alpha^{12}
 \end{aligned}$$

$$S(x) = x + x^2 + \alpha^{10}x^3 + x^4 + \alpha^{10}x^5 + \alpha^5x^6$$

Applying Berlekamp's algorithm, we obtain the following

n	$\Lambda(x)^{(2n)}$	$B(x)^{(2n)}$	$\Delta^{(2n)}$
0	1	1	1
1	$1 + x$	x	α^5
2	$1 + x + \alpha^5 x^2$	$x(1 + x)/\alpha^5$	α^{10}
3	$1 + x + \alpha^5 x^3$

The error locator polynomial is then

$$\Lambda(x) = 1 + x + \alpha^5 x^3 = (1 + \alpha^3 x)(1 + \alpha^5 x)(1 + \alpha^{12} x)$$

Indicating errors at the positions corresponding to α^3 , α^5 and α^{12} . The corrected received word with the corrected positions is

$$c = (0000000000000000) \\ \downarrow \\ c(x) = 0$$

Example 4.22 Let the transmission code be the double-error-correcting, narrow-sense, binary BCH code of length 15. The generator polynomial is $g(x)$. Use Berlekamp's algorithm to decode the following received vector = (000110001100000).

Solution For double error correction, the generator polynomial $g(x) = 1 + x^4 + x^6 + x^7 + x^8$ has roots which include four consecutive powers of α : $(\alpha, \alpha^2, \alpha^3, \alpha^4)$ where α is primitive in GF(16). The received vector is

$$r = (000110001100000) \\ \downarrow \\ r(x) = x^3 + x^4 + x^8 + x^9$$

The syndrome polynomial is written as

$$S(x) = S_1x + S_2x^2 + S_3x^3 + S_4x^4 \\ S_1 = r(\alpha) = \alpha^3 + \alpha^4 + \alpha^8 + \alpha^9 = \alpha^2 \\ S_2 = r(\alpha^2) = \alpha^6 + \alpha^8 + \alpha^{16} + \alpha^{18} = \alpha^4 \\ S_3 = r(\alpha^3) = \alpha^9 + \alpha^{12} + \alpha^{24} + \alpha^{27} = 0 \\ S_4 = r(\alpha^4) = \alpha^{12} + \alpha^{16} + \alpha^{32} + \alpha^{36} = \alpha^8$$

Since $\alpha^{15} = 1$;

$$\begin{aligned}\alpha^{16} &= \alpha^{15}\alpha^1 = \alpha^1; \alpha^{18} = \alpha^{15}\alpha^3 = \alpha^3; \alpha^{24} = \alpha^{15}\alpha^9 = \alpha^9 \\ \alpha^{28} &= \alpha^{15}\alpha^{12} = \alpha^{12}; \alpha^{32} = \alpha^{15}\alpha^{15}\alpha^2 = \alpha^2; \alpha^{36} = \alpha^{15}\alpha^{15}\alpha^6 = \alpha^6 \\ S(x) &= \alpha^2x + \alpha^4x^2 + \alpha^8x^4\end{aligned}$$

Applying Berlekamp's algorithm, we obtain the following

n	$\Lambda(x)^{(2n)}$	$B(x)^{(2n)}$	$\Delta^{(2n)}$
0	1	1	α^2
1	$1 + \alpha^2x$	$\alpha^{13}x$	α^6
2	$1 + \alpha^2x + \alpha^{19}x^2$

The error locator polynomial is then

$$\Lambda(x) = 1 + \alpha^2x + \alpha^{19}x^2 = (1 + \alpha^7x)(1 + \alpha^{12}x)$$

Indicating errors at the positions corresponding to α^7 and α^{12} . The corrected received word with the corrected positions is

$$\begin{aligned}c &= (000110011100100) \\ c(x) &= x^3 + x^4 + \overset{\downarrow}{x^7} + x^8 + x^9 + x^{12}\end{aligned}$$

4.5 Reed–Solomon Codes

The Reed–Solomon (RS) codes are the most powerful non-binary block codes which have seen widespread applications. These codes work with symbols that consist of several bits. A common symbol size for non-binary codes is 8 bits or a byte. The RS codes are good at correcting burst errors because the correction of these codes is done on the symbol level.

A given Reed–Solomon code is indicated by referring to it as an (n, k) code. The parameter n indicates the codeword length in terms of the number of symbols in the codeword. The parameter k indicates the number of message symbols in the codeword. The number of parity symbols added is thus $(n - k)$. The error-correcting capability of the code is $t_{ec} = (n - k)/2$. The minimum distance of Reed–Solomon code is $(n - k + 1)$.

4.5.1 Reed–Solomon Encoder

Generator Polynomial

A general form of the polynomial $g(x)$ used in RS code generation is

$$g(x) = (x - \alpha^i)(x - \alpha^{i+1}) \dots (x - \alpha^{i+2t_{ec}}) \quad (4.35)$$

where α is a primitive element of the Galois field.

The codeword $c(x)$ is constructed using

$$c(x) = g(x) \cdot i(x) \quad (4.36)$$

where $i(x)$ is the information polynomial.

The code word $c(x)$ is exactly divisible by the generator polynomial $g(x)$. The remainder obtained by dividing $i(x) \cdot x^{n-k}$ by $g(x)$ gives the parity polynomial $p(x)$ as

$$p(x) = i(x) \cdot x^{n-k} / g(x) \quad (4.37)$$

The parity symbols are computed by performing a polynomial division using GF algebra. The steps involved in this computation are as follows:

- Step 1. Multiply the message symbols by x^{n-k} (This shifts the message symbols to the left to make room for the $(n - k)$ parity symbols).
- Step 2. Divide the message polynomial by the code generator polynomial using GF algebra.
- Step 3. The parity symbols are the remainder of this division. These steps are accomplished in hardware using a shift register with feedback. The architecture for the encoder is shown in Fig. 4.9.

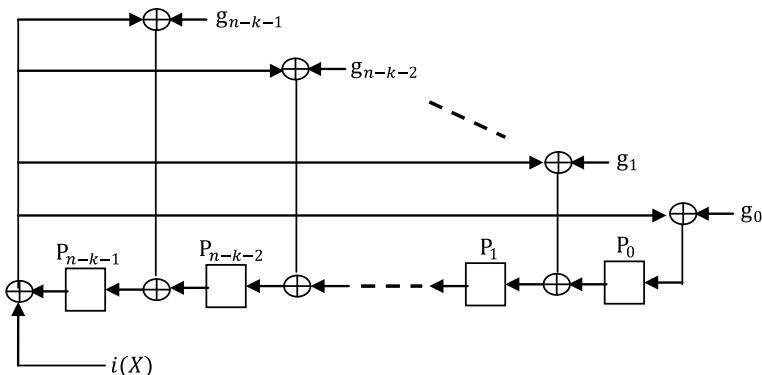


Fig. 4.9 Reed–Solomon encoder

$g(x)$ is the generator polynomial used to generate parity symbols $p(x)$. The number of registers used is equal to $n - k$. Parity symbols are generated by serial entry of the information symbols into $i(x)$.

The resultant code word is given by

$$c(x) = i(x) \cdot x^{n-k} + p(x) \quad (4.38)$$

Example 4.23 Construct a generator polynomial for a (15, 11) Reed–Solomon code with elements in $\text{GF}(2^4)$.

Solution A (15, 11) Reed–Solomon code has minimum distance 5. Thus, the (15, 11) Reed–Solomon code is double error corrections. It must have 4 consecutive powers of α as zeros.

The generator polynomial is constructed as follows using the representation for $\text{GF}(16)$ over $\text{GF}(2)$.

$$\begin{aligned} g(x) &= (x - \alpha)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4) \\ &= (x^2 + (\alpha^2 + \alpha)x + \alpha^3)(x^2 + (\alpha^3 + \alpha^4)x + \alpha^7) \\ &= (x^2 + \alpha^5x + \alpha^3)(x^2 + \alpha^7x + \alpha^7) \\ &= (x^4 + (\alpha^5 + \alpha^7)x^3 + (\alpha^3 + \alpha^{12} + \alpha^7)x^2 + (\alpha^{10} + \alpha^{12})x + \alpha^{10}) \\ &= (x^4 + \alpha^{13}x^3 + \alpha^6x^2 + \alpha^3x + \alpha^{10}) \end{aligned}$$

Example 4.24 Compute a generator polynomial for a double-error-correcting Reed–Solomon code of length 31.

Solution Let α be a root of the primitive binary polynomial $x^5 + x^2 + 1$ and thus a primitive 31st of unity. The resulting code is to be a double-error-correcting code, and it must have 4 consecutive powers of α as zeros. A narrow-sense generator is constructed as follows using the representation for $\text{GF}(32)$.

$$\begin{aligned} g(x) &= (x - \alpha)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4) \\ &= (x^2 + (\alpha^2 + \alpha)x + \alpha^3)(x - \alpha^3)(x - \alpha^4) \\ &= (x^2 + \alpha^{19}x + \alpha^3)(x - \alpha^3)(x - \alpha^4) \\ &= (x^3 + \alpha^{19}x^2 + \alpha^3x^2 + \alpha^3x + \alpha^{22}x + \alpha^6)(x - \alpha^4) \\ &= (x^3 + (\alpha^{19} + \alpha^3)x^2 + (\alpha^3 + \alpha^{22})x + \alpha^6)(x - \alpha^4) \\ &= (x^3 + \alpha^{12}x^2 + \alpha^{14}x + \alpha^6)(x - \alpha^4) \\ &= x^4 + \alpha^{12}x^3 + \alpha^4x^3 + \alpha^{14}x^2 + \alpha^{16}x^2 + \alpha^6x + \alpha^{18}x + \alpha^{10} \\ &= x^4 + (\alpha^{12} + \alpha^4)x^3 + (\alpha^{14} + \alpha^{16})x^2 + (\alpha^6 + \alpha^{18})x + \alpha^{10} \\ &= x^4 + \alpha^{24}x^3 + \alpha^{19}x^2 + \alpha^{29}x + \alpha^{10} \end{aligned}$$

Example 4.25 Compute a generator polynomial for a triple-error-correcting Reed–Solomon code of length 15.

Solution

$$\begin{aligned}
 g(x) &= (x + \alpha)(x + \alpha^2)(x + \alpha^3)(x + \alpha^4)(x + \alpha^5)(x + \alpha^6) \\
 &= (x^2 + (\alpha^2 + \alpha)x + \alpha^3)(x^2 + (\alpha^3 + \alpha^4)x + \alpha^7) \\
 &\quad (x^2 + (\alpha^6 + \alpha^5)x + \alpha^{11}) \\
 &= (x^2 + \alpha^5x + \alpha^3)(x^2 + \alpha^7x + \alpha^7)(x^2 + \alpha^9x + \alpha^{11}) \\
 &= (x^4 + (\alpha^5 + \alpha^7)x^3 + (\alpha^3 + \alpha^{12} + \alpha^7)x^2 + (\alpha^{10} + \alpha^{12})x + \alpha^{10}) \\
 &\quad (x^2 + \alpha^9x + \alpha^{11}) \\
 &= (x^4 + \alpha^{13}x^3 + \alpha^6x^2 + \alpha^3x + \alpha^{10})(x^2 + \alpha^9x + \alpha^{11}) \\
 &= (x^6 + (\alpha^9 + \alpha^{13})x^5 + (\alpha^{11} + \alpha^{22} + \alpha^6)x^4 + (\alpha^{24} + \alpha^{15} + \alpha^3)x^3 \\
 &\quad + (\alpha^{10} + \alpha^{17} + \alpha^{12})x^2 + (\alpha^{14} + \alpha^{19})x + \alpha^{21}) \\
 &= (x^6 + \alpha^{10}x^5 + \alpha^{14}x^4 + \alpha^4x^3 + \alpha^6x^2 + \alpha^9x + \alpha^6)
 \end{aligned}$$

Basic Properties of Reed–Solomon Codes

1. Non-binary BCH codes are referred to as Reed–Solomon codes.
2. The minimum distance of Reed–Solomon code is $(n - k + 1)$.
3. RS codes are maximum distance separable (MDS). The singleton bound implies that $d_{\min} \leq (n - k + 1)$. RS (n, k) code is called MDS if the singleton bound is satisfied with equality.
4. The weight distribution polynomial of RS code is known. The weight distribution of an RS code with symbols from $\text{GF}(q)$ and with block length $n = q - 1$ and minimum distance d_{\min} is given by

$$W_i = \binom{n}{i} n \sum_{j=0}^{1-d_{\min}} (-1)^j \binom{i-1}{j} (n+1)^{i-j-d_{\min}} d_{\min} \leq i \leq n \quad (4.39)$$

4.5.2 Decoding of Reed–Solomon Codes

The locations of the errors can be found from the error locator polynomial $\Lambda(x)$. Once the locations of the errors are known, the magnitudes of the errors are found by the Forney's algorithm given by [1]

$$e_k = \frac{-x_k \Omega(x_k^{-1})}{\Lambda'(x_k^{-1})} \quad (4.40)$$

where e_k represents the error magnitude at the k th location, and $\Lambda'(x_k)$ stands for formal derivative of the error locator polynomial $\Lambda(x)$. If locator polynomial $\Lambda(x) = \Lambda_v x^v + \Lambda_{v-1} x^{v-1} + \dots + \Lambda_1 x + 1$ is a polynomial with coefficients in $\text{GF}(q)$, the formal derivative $\Lambda'(x)$ is defined as

$$\Lambda'(x) = v\Lambda_v x^{v-1} + (v-1)\Lambda_{v-1} x^{v-2} + \dots + \Lambda_1 \quad (4.41)$$

The decoding of a RS code has to go through the following six steps:

- Step 1. Compute syndromes from the received polynomial $r(x)$.
- Step 2. Apply Berlekamp–Massey algorithm to compute error location polynomial $\Lambda(x)$.
- Step 3. Compute error magnitude polynomial $\Omega(x) = \Lambda(x)(1 + S(x)) \bmod x^{2t_{ec}+1}$.
- Step 4. Find the roots of $\Lambda(x)$, the inverse of the roots indicates the locations of the errors.
- Step 5. Compute the error magnitudes and determine the error polynomial $e(x)$.
- Step 6. Subtract $e(x)$ from the received polynomial to correct the errors.

Syndrome generation is similar to parity calculation. A Reed–Solomon codeword has $2t_{ec}$ syndromes that depend only on errors (not on the transmitted codeword).

The syndrome sequence can be computed for the received word polynomial $r(x)$ by substituting the $2t_{ec}$ roots of the generator polynomial $g(x)$ into $r(x)$. The Berlekamp–Massey algorithm or Euclid's algorithm can be used to find error locator polynomial. The Euclid's algorithm is widely used in practice as it is easy for implementation. However, hardware and software implementations of the Berlekamp–Massey algorithm are more efficient [2, 3]. Once the error locator polynomial is known, the error locations can be found by using the Chien search algorithm [4].

The Berlekamp–Massey Decoding Algorithm.

The problem of decoding RS codes can be viewed as finding a linear feedback shift register (LFSR) of minimal length so that the first $2t_{ec}$ elements in the LFSR output sequence are the syndromes $S_1 S_2 \dots S_{2t_{ec}}$. The error locator polynomial $\Lambda(x)$ is provided by the taps of the LFSR.

The flowchart of the Berlekamp–Massey iterative algorithm is shown in Fig. 4.10. Here, $\Lambda^{(n)}(x)$ is the error location polynomial at the n th iteration step, $B(x)$ stands for the connection polynomial, L_n represents the length of LFSR at index n , and d_n is the discrepancy. Consider the error location polynomial $\Lambda^{(n)}(x)$ of length n . The coefficients of the polynomial specify the taps of a length- n LFSR. The Berlekamp–Massey algorithm initially (i.e., $n = 0$) sets the tap coefficient and the length of the LFSR to 1 and 0, respectively, to indicate that the computed error locator polynomial $\Lambda^{(0)}(x)$ and its length are set to 1 and 0, respectively, and also sets $B(x) = x$ at every iteration, or a new syndrome component, the discrepancy d_n is computed by

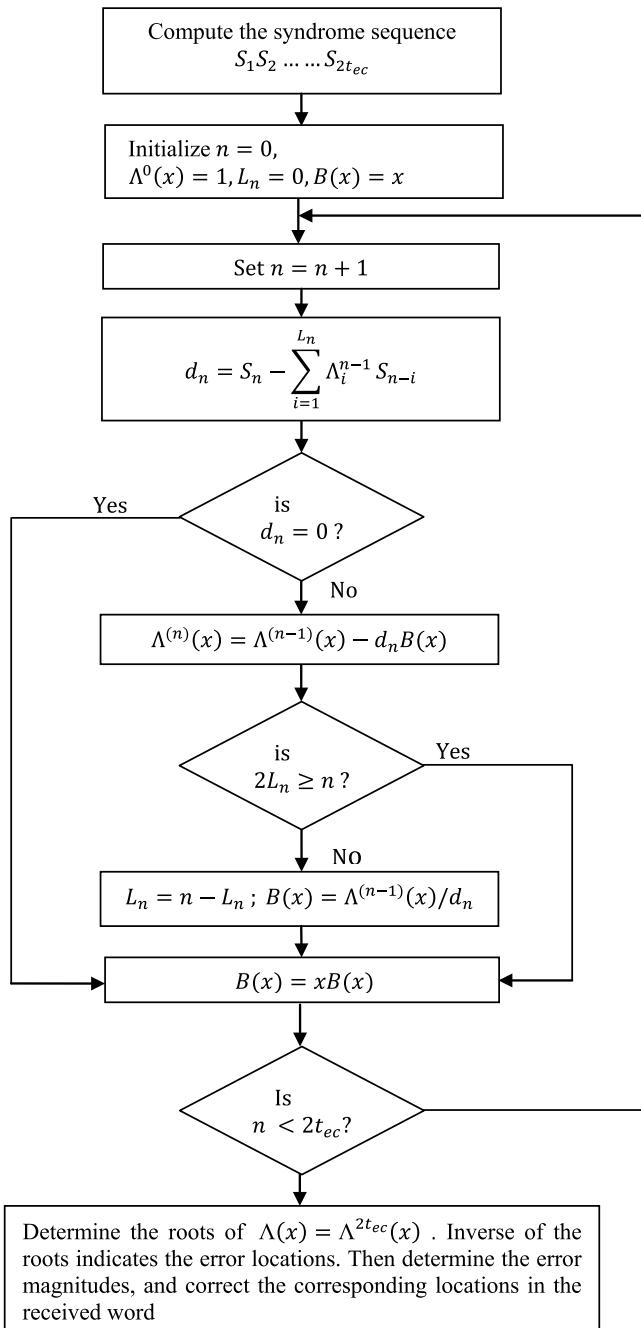


Fig. 4.10 Berlekamp–Massey iterative algorithm

subtracting the n th output of the LFSR defined by $\Lambda^{(n-1)}(x)$ from the n th syndrome. If the discrepancy is not equal to zero, a modified error locator polynomial is constructed using discrepancy and connection polynomial $B(x)$. Then, the length of the LFSR is to be tested. If $2L_n$ is greater than or equal to n , the length of the LFSR and connection polynomial $B(x)$ are to be updated. Otherwise, if $2L_n$ is less than n , the connection polynomial $B(x)$ is to be reset as $xB(x)$.

If the discrepancy is equal to zero, then the connection polynomial $B(x)$ is to be reset as $xB(x)$ and the previous error locator polynomials are used for the next iteration. The process is continued, and the algorithm stops at the end of the iteration $n = 2t_{ec}$ and $\Lambda^{(2t_{ec})}(x)$ is taken as the error locator polynomial $\Lambda(x)$.

Example 4.26 Let the transmission code be the double-error-correcting RS code of length 7. Use the Berlekamp–Massey algorithm to decode the following received vector $r = (00\alpha^5 1\alpha^2 0\alpha^2)$

Solution

Step 1: The received polynomial is

$$r(x) = \alpha^5 x^2 + x^3 + \alpha^2 x^4 + \alpha^2 x^6; \quad \text{i.e., } r = (00\alpha^5 1\alpha^2 0\alpha^2)$$

For double-error-correcting code, the syndrome polynomial is

$$S(x) = S_1 x + S_2 x^2 + S_3 x^3 + S_4 x^4$$

The syndromes S_1, S_2, S_3 and S_4 for the above-mentioned received polynomial are computed using the representation for GF(8) as

$$S_1 = r(\alpha) = \alpha^6$$

$$S_2 = r(\alpha^2) = \alpha^3$$

$$S_3 = r(\alpha^3) = \alpha^4$$

$$S_4 = r(\alpha^4) = \alpha^3$$

Thus, the

$$S(x) = \alpha^6 x + \alpha^3 x^2 + \alpha^4 x^3 + \alpha^3 x^4.$$

Step 2: Berlekamp–Massey algorithm proceeds as follows:

n	S_n	$\Lambda^{(n)}(x)$	d_n	L_n	$B(x)$
0	...	1	...	0	x
1	α^6	$1 + \alpha^6x$	$S_1 - 0 = \alpha^6$	1	αx
2	α^3	$1 + (\alpha^6 + \alpha^3)x$ $= 1 + \alpha^4x$	$S_2 - \alpha^6\alpha^6$ $= S_2 - \alpha^5 = \alpha^2$	1	αx^2
3	α^4	$1 + \alpha^4x + \alpha^5\alpha x^2$ $= 1 + \alpha^4x + \alpha^6x^2$	$S_3 - \alpha^4\alpha^3$ $= S_3 - 1 = \alpha^5$	2	$\left(\frac{1 + \alpha^4x}{\alpha^5}\right)x$ $= (\alpha^2x + \alpha^6x^2)$
4	α^3	$1 + \alpha^2x + \alpha x^2$	$S_4 - (\alpha^4\alpha^4 + \alpha^6\alpha^3)$ $= S_4 - (\alpha + \alpha^2)$ $= S_4 - \alpha^4 = \alpha^6$

The error locator polynomial is then

$$\Lambda(x) = 1 + \alpha^2x + \alpha x^2$$

Step 3: The error magnitude polynomial is

$$\begin{aligned} \Omega(x) &= \Lambda(x)(1 + S(x)) \bmod x^{2t_{ec}+1} \\ &= (1 + \alpha^2x + \alpha x^2)(1 + \alpha^6x + \alpha^3x^2 + \alpha^4x^3 + \alpha^3x^4) \bmod x^5 \\ &= 1 + x + \alpha^3x^2 \end{aligned}$$

Step 4:

$$\Lambda(x) = 1 + \alpha^2x + \alpha x^2 = (1 + \alpha^3x)(1 + \alpha^5x) = 0$$

The factorization of the error locator polynomial indicates that there are errors in the third and fifth positions of the received vector.

Hence, the error polynomial $e(x)$ is

$$e(x) = e_3x^3 + e_5x^5$$

Step 5: From the error locator polynomial, it is known that error positions are in locations 3 and 5. Now, the error magnitudes can be computed by using error evaluator polynomial $\Omega(x)$ and derivative of the error locator polynomial $\Lambda(x)$. The error magnitudes are given by

$$e_k = \frac{-x_k \Omega(x_k^{-1})}{\Lambda'(x_k^{-1})}$$

The magnitudes of errors are found to be

$$e_3 = \frac{-x_3 \Omega(x_3^{-1})}{\Lambda'(x_3^{-1})}$$

Since $\Lambda'(x_3^{-1}) = \alpha^2$

$$e_3 = \frac{-x_3(1 + x_3^{-1} + \alpha^3 x_3^{-2})}{\alpha^2}$$

where $x_3 = \alpha^3$

Thus,

$$e_3 = \frac{(\alpha^3 + 1 + 1)}{\alpha^2} = \alpha$$

Similarly,

$$e_5 = \frac{-x_5(1 + x_5^{-1} + \alpha^3 x_5^{-2})}{\alpha^2}$$

where $x_5 = \alpha^5$

Hence,

$$e_5 = \frac{(\alpha^5 + 1 + \alpha^{-2})}{\alpha^2} = \alpha^5$$

Thus, the error pattern

$$e(x) = \alpha x^3 + \alpha^5 x^5$$

Step 6:

$$c(x) = r(x) - e(x) = \alpha^5 x^2 + x^3 + \alpha^2 x^4 + \alpha^2 x^6 + \alpha x^3 + \alpha^5 x^5$$

Example 4.27 Consider a triple-error-correcting Reed–Solomon code of length 15. Decode the received vector $r = (000\alpha^7 00\alpha^3 00000\alpha^4 00)$ using Berlekamp–Massey algorithm.

Solution

Step 1: The received polynomial is

$$r(x) = \alpha^7x^3 + \alpha^3x^6 + \alpha^4x^{12}; \quad \text{i.e., } r = (000\alpha^700\alpha^300000\alpha^400)$$

The following syndromes are computed using the representation of GF(16) over GF(2). For triple error correction, the roots of the generator polynomial include $\alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6$.

Thus

$$S_1 = r(\alpha) = \alpha^{10} + \alpha^9 + \alpha = \alpha^{12}$$

$$S_2 = r(\alpha^2) = \alpha^{13} + 1 + \alpha^{13} = 1$$

$$S_3 = r(\alpha^3) = \alpha + \alpha^6 + \alpha^{10} = \alpha^{14}$$

$$S_4 = r(\alpha^4) = \alpha^4 + \alpha^{12} + \alpha^7 = \alpha^{10}$$

$$S_5 = r(\alpha^5) = \alpha^7 + \alpha^3 + \alpha^4 = 0$$

$$S_6 = r(\alpha^6) = \alpha^{10} + \alpha^9 + \alpha = \alpha^{12}$$

$$S(x) = \alpha^{12}x + x^2 + \alpha^{14}x^3 + \alpha^{10}x^4 + \alpha^{12}x^6.$$

Step 2: Berlekamp–Massey algorithm proceeds as follows:

n	S_n	$\Lambda^{(n)}(x)$	d_n	L_n	$B(x)$
0	...	1	...	0	x
1	α^{12}	$1 + \alpha^2x$	$S_1 - 0 = \alpha^2$	1	α^3x
2	1	$1 + \alpha^3x$	$S_2 - \alpha^9 = \alpha^7$	1	α^3x^2
3	α^{14}	$1 + \alpha^3x + \alpha^3x^2$	$S_3 - \alpha^3 = 1$	2	$x + \alpha^3x^2$
4	α^{10}	$1 + \alpha^4x + \alpha^{12}x^2$	$S_4 - \alpha^6 = \alpha^7$	2	$x^2 + \alpha^3x^3$
5	0	$1 + \alpha^4x + \alpha^3x^2 + \alpha^{13}x^3$	$S_5 - \alpha^{10} = \alpha^{10}$	3	$\alpha^5x + \alpha^9x^2 + \alpha^2x^3$
6	α^{12}	$1 + \alpha^7x + \alpha^4x^2 + \alpha^6x^3$	$S_6 - \alpha = \alpha^{13}$

The error locator polynomial is then

$$\Lambda(x) = 1 + \alpha^7x + \alpha^4x^2 + \alpha^6x^3 = (1 + \alpha^3x)(1 + \alpha^6x)(1 + \alpha^{12}x)$$

Step 3: The error magnitude polynomial is

$$\begin{aligned} \Omega(x) &= \Lambda(x)(1 + S(x)) = (1 + \alpha^7x + \alpha^4x^2 + \alpha^6x^3)(1 + \alpha^{12}x + x^2 \\ &\quad + \alpha^{14}x^3 + \alpha^{10}x^4 + \alpha^{12}x^6) \bmod x^7 \end{aligned}$$

$$\begin{aligned}
 &= (1 + \alpha^2x + x^2 + \alpha^6x^3 + x^7 + \alpha x^8 + \alpha^3x^9) \bmod x^7 \\
 &= (1 + \alpha^2x + x^2 + \alpha^6x^3)
 \end{aligned}$$

Step 4:

$$\Lambda(x) = 1 + \alpha^7x + \alpha^4x^2 + \alpha^6x^3 = (1 + \alpha^3x)(1 + \alpha^6x)(1 + \alpha^{12}x) = 0$$

The factorization of the error locator polynomial indicates that there are errors in the positions 3, 6, and 12 of the received vector.

Hence, the error polynomial $e(x)$ is

$$e(x) = e_{12}x^{12} + e_6x^6 + e_3x^3$$

Step 5: From the error locator polynomial, it is known that error positions are at locations 3, 6, and 12. Now, the error magnitudes can be computed by using error evaluator polynomial $\Omega(x)$ and derivative of the error locator polynomial $\Lambda(x)$. The error magnitudes are given by

$$e_k = \frac{-x_k \Omega(x_k^{-1})}{\Lambda'(x_k^{-1})}$$

The magnitudes of errors are found to be

$$e_3 = \frac{-x_3 \Omega(x_3^{-1})}{\Lambda'(x_3^{-1})}$$

$$\text{Since } \Lambda'(x_3^{-1}) = \alpha^7 + \alpha^6x_3^{-2}$$

$$e_3 = \frac{-x_3(1 + \alpha^2x_3^{-1} + x_3^{-2} + \alpha^6x_3^{-3})}{\alpha^7 + \alpha^6x_3^{-2}}$$

$$\text{where } x_3 = \alpha^3$$

Thus,

$$e_3 = \frac{\alpha^3(1 + \alpha^2 \cdot \alpha^{12} + \alpha^9 + \alpha^{12})}{1 + \alpha^7} = \frac{\alpha^3(1 + \alpha^{14} + \alpha^9 + \alpha^{12})}{1 + \alpha^7} = \frac{\alpha^3 \cdot \alpha^{13}}{\alpha^9} = \alpha^7$$

Similarly,

$$e_6 = \alpha^3; e_{12} = \alpha^4.$$

Thus, the error pattern

$$e(x) = \alpha^7x^3 + \alpha^3x^6 + \alpha^4x^{12}$$

Step 6:

The corrected received word is $c(x) = r(x) - e(x) = 0$

$$c = (0000000000000000)$$

Example 4.28 Let the transmission code be the triple-error-correcting RS code of length 31. Decode the received vector $r = (00\alpha^800\alpha^20000\alpha000000000000000000000000000000)$ using Berlekamp–Massey algorithm.

Solution

Step 1: The received polynomial is $r(x) = \alpha^8x^2 + \alpha^2x^5 + \alpha x^{10}$; i.e, $r = (00\alpha^800\alpha^20000\alpha000000000000000000000000000000)$

The following syndromes are computed using the representation of GF(16) over GF(2). For triple error correction, the roots of the generator polynomial include $\alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6$.

Thus

$$S_1 = r(\alpha) = \alpha^{10} + \alpha^9 + \alpha^{11} = \alpha$$

$$S_2 = r(\alpha^2) = \alpha^{12} + \alpha^{12} + \alpha^{21} = \alpha^{21}$$

$$S_3 = r(\alpha^3) = \alpha^{14} + \alpha^{17} + \alpha^{31} = \alpha^{23}$$

$$S_4 = r(\alpha^4) = \alpha^{16} + \alpha^{22} + \alpha^{20} = \alpha^{15}$$

$$S_5 = r(\alpha^5) = \alpha^{18} + \alpha^{27} + \alpha^{20} = \alpha^2$$

$$S_6 = r(\alpha^6) = \alpha^{20} + \alpha + \alpha^{30} = \alpha^{13}$$

$$S(x) = \alpha x + \alpha^{21}x^2 + \alpha^{23}x^3 + \alpha^{15}x^4 + \alpha^2x^5 + \alpha^{13}x^6.$$

Step 2: Berlekamp–Massey algorithm proceeds as follows:

n	s_n	$\Lambda^{(n)}(x)$	d_n	L_n	$B(x)$
0	\dots	1	\dots	0	x
1	α	$1 + \alpha x$	$S_1 - 0 = \alpha$	1	$\alpha^{30}x$
2	α^{21}	$1 + \alpha^{20}x$	$S_2 - \alpha^2 = \alpha^{13}$	1	$\alpha^{30}x^2$
3	α^{23}	$1 + \alpha^{20}x + \alpha^{23}x^2$	$S_3 - \alpha^{10} = \alpha^{24}$	2	$\alpha^7x + \alpha^{27}x^2$
4	α^{15}	$1 + \alpha^{20}x + \alpha^{23}x^2 + \alpha^{15}x + \alpha^4x^2$ $= 1 + \alpha^{17}x + \alpha^{15}x^2$	$S_4 - \alpha^{12} - \alpha^{13} = \alpha^8$	2	$\alpha^7x^2 + \alpha^{27}x^3$
5	α^2	$1 + \alpha^{17}x + \alpha^{22}x^2 + \alpha^{26}x^3$	$S_5 - \alpha - \alpha^7 = \alpha^{30}$	3	$\alpha^{16}x^3 + \alpha^{18}x^2 + \alpha x$
6	α^{13}	$1 + \alpha^{17}x + \alpha^{22}x^2 + \alpha^{26}x^3 + \alpha^2x^3 + \alpha^{18}x + \alpha^4x^2$ $= 1 + \alpha^4x + \alpha^5x^2 + \alpha^{17}x^3$	$S_6 - \alpha^{19} - \alpha^6 - \alpha^{18} = \alpha^{17}$ \dots	\dots	

The error locator polynomial is then

$$\Lambda(x) = 1 + \alpha^4x + \alpha^5x^2 + \alpha^{17}x^3 = (1 + \alpha^{21}x)(1 + \alpha^{26}x)(1 + \alpha^{29}x)$$

Step 3: The error magnitude polynomial is

$$\begin{aligned}\Omega(x) &= \Lambda(x)(1 + S(x)) = (1 + \alpha^4x + \alpha^5x^2 + \alpha^{17}x^3)(1 + \alpha x + \alpha^{21}x^2 + \alpha^{23}x^3 \\ &\quad + \alpha^{15}x^4 + \alpha^2x^5 + \alpha^{13}x^6) \bmod x^7 \\ &= (1 + \alpha^{30}x + \alpha^{21}x^2 + \alpha^{23}x^3)\end{aligned}$$

Step 4:

$$\Lambda(x) = 1 + \alpha^4x + \alpha^5x^2 + \alpha^{17}x^3 = (1 + \alpha^2x)(1 + \alpha^5x)(1 + \alpha^{10}x) = 0$$

The factorization of the error locator polynomial indicates that there are errors in the second, fifth, and tenth positions of the received vector.

Hence, the error polynomial $e(x)$ is

$$e(x) = e_{10}x^{10} + e_5x^5 + e_2x^2$$

Step 5: From the error locator polynomial, it is known that error positions are at locations 2, 5, and 10. Now, the error magnitudes can be computed by using error evaluator polynomial $\Omega(x)$, and derivative of the error locator polynomial $\Lambda(x)$. The error magnitudes are given by

$$e_k = \frac{-x_k \Omega(x_k^{-1})}{\Lambda'(x_k^{-1})}$$

The magnitudes of errors are found to be

$$e_2 = \frac{-x_2 \Omega(x_2^{-1})}{\Lambda'(x_2^{-1})}$$

Since $\Lambda'(x_2^{-1}) = \alpha^4 + \alpha^{17}x_2^{-2}$

$$e_2 = \frac{-x_2(1 + \alpha^2x_2^{-1} + x_2^{-2} + \alpha^6x_2^{-3})}{\alpha^4 + \alpha^{17}x_2^{-2}}$$

where $x_3 = \alpha^3$

Thus,

$$e_2 = \frac{\alpha^2(1 + \alpha^{30} \cdot \alpha^{-2} + \alpha^{21} \cdot \alpha^{-4} + \alpha^{23} \cdot \alpha^{-6})}{\alpha^4 + \alpha^{13}} = \frac{\alpha^2 + \alpha^{30}}{\alpha^{20}} = \frac{\alpha^{28}}{\alpha^{20}} = \alpha^8$$

Similarly,

$$e_5 = \alpha^2; e_{10} = \alpha.$$

Thus, the error pattern

$$e(x) = \alpha^8x^2 + \alpha^7x^5 + \alpha x^{10}$$

Step 6: The corrected received word is $c(x) = r(x) - e(x) = 0$

$$c = (0000000000000000)$$

4.5.3 Binary Erasure Decoding

For binary linear codes, erasure decoding is done by the following three steps:

- Step 1. Replace all erasures with zeros in a received word and decode it to a codeword c_0 .
- Step 2. Replace all erasures with ones in a received word and decode it to a codeword c_1 .
- Step 3. Choose the final code word either c_0 or c_1 that is closest to the received word in the Hamming distance.

4.5.4 Non-binary Erasure Decoding

Suppose that a received word has v errors and f erasures. An erasure locator polynomial can be written as

$$\Gamma(x) = \prod_{l=1}^f (1 - Y_l x) \quad (4.42)$$

where Y_l stands for erasure locators. Now, the decoding has to find out error locations and compute the error magnitudes of the error locators and erasure magnitudes of the erasure locators. To find the error locator polynomial, a modified syndrome polynomial is to be formulated and Berlekamp–Massey algorithm is to be applied on the modified syndrome coefficients.

The modified syndrome polynomial is given by

$$S^M(x) \equiv (\Gamma(x)[1 + S(x)] - 1) \bmod x^{2t+1} \quad (4.43)$$

where the coefficients of the syndrome polynomial $S(x)$ are computed using the following

$$S_l = r(\alpha^l) \quad (4.44)$$

replacing all the erasures with zeros in the received polynomial $r(x)$.

After finding the error locator polynomial $\Lambda(x)$, obtain error magnitude polynomial and error/erasure locator polynomial as

$$\Omega(x) = \Lambda(x)[1 + S^M(x)]x^{2t+1} \quad (4.45)$$

$$\psi(x) = \Lambda(x)\Gamma(x) \quad (4.46)$$

Then, using the modified Forney's algorithm, compute the error and erasure magnitudes as given by

$$e_k = \frac{-X_k \Omega(X_k^{-1})}{\Psi'(X_k^{-1})} \quad (4.47a)$$

$$f_k = \frac{-Y_k \Omega(Y_k^{-1})}{\Psi'(Y_k^{-1})} \quad (4.47b)$$

Knowing the magnitudes of the error locators and erasure locators, an error/erasure polynomial can be constructed and subtracted from the received polynomial to arrive at the desired code polynomial.

The stepwise procedure using Berlekamp–Massey algorithm for error/erasure decoding is as follows:

- Step 1. Formulate the erasure polynomial $\Gamma(x)$ using the erasures in the received vector.
- Step 2. Obtain the syndrome polynomial $S(x)$ replacing the erasures with zeros.
- Step 3. Compute the modified syndrome polynomial using Eq. (4.43).
- Step 4. Apply the Berlekamp–Massey on modified syndrome coefficients to find the error correction polynomial $\Lambda(x)$.
- Step 5. Find the roots of $\Lambda(x)$, to determine the error locations.

- Step 6. Compute the error magnitudes using Eq. (4.47a) and determine the error polynomial $e(x)$.
- Step 7. Compute the erasure magnitudes using Eq. (4.47b) and determine the erasure polynomial $f(x)$.
- Step 8. Subtract $e(x)$ and $f(x)$ from the received polynomial to correct the errors.

Example 4.29 Let the transmission code be the double-error-correcting RS code of length 7. Use the Berlekamp–Massey algorithm to decode the following received vector $r = (00\alpha^301f1)$.

Solution

Step 1: The received polynomial is $r(x) = \alpha^3x^2 + x^4 + fx^5 + x^6$; The f indicates an erasure. This erasure gives the erasure polynomial

$$\Gamma(x) = 1 + \alpha^5x$$

Step 2: Place a zero in the erasure location and compute the syndromes. For double-error-correcting code, the syndrome polynomial is

$$\begin{aligned} S(x) &= S_1x + S_2x^2 + S_3x^3 + S_4x^4 \\ S_l &= \alpha^3(\alpha^l)^2 + (\alpha^l)^4 + (\alpha^l)^6 \end{aligned}$$

The syndromes S_1, S_2, S_3 and S_4 for the above-mentioned received polynomial are computed using the representation for GF(8) as

$$S_1 = r(\alpha) = \alpha^5 + \alpha^4 + \alpha^6 = \alpha^2$$

$$S_2 = r(\alpha^2) = \alpha^7 + \alpha^8 + \alpha^{12} = \alpha^2$$

$$S_3 = r(\alpha^3) = \alpha^9 + \alpha^{12} + \alpha^{18} = \alpha^6$$

$$S_4 = r(\alpha^4) = \alpha^{11} + \alpha^{16} + \alpha^{24} = \alpha^4$$

Thus the

$$S(x) = \alpha^2x + \alpha^2x^2 + \alpha^6x^3 + x^4.$$

Step 3: Compute the modified syndrome polynomial,

$$\begin{aligned} 1 + S^M(x) &\equiv \Gamma(x)[1 + S(x)] \bmod x^{2t+1} \\ &\equiv (1 + \alpha^5x)(1 + \alpha^2x + \alpha^2x^2 + \alpha^6x^3 + x^4) \bmod x^5 \\ &\equiv 1 + \alpha^3x + \alpha^6x^2 + \alpha^2x^3 + \alpha^5x^4 \bmod x^5 \end{aligned}$$

$S^M(x)$ is thus $\alpha^3x + \alpha^6x^2 + \alpha^2x^3 + \alpha^5x^4$.

Step 4: Berlekamp–Massey algorithm proceeds as follows:

n	S_n^M	$\Lambda^{(n)}(x)$	d_n	L_n	$B(x)$
0	...	1	...	0	x
1	α^3	$1 + \alpha^3x$	α^3	1	α^4x
2	α^6	$1 + \alpha^3x$	0	1	α^4x^2
3	α^2	$1 + \alpha^3x$	0	1	α^4x^3
4	α^5	$1 + \alpha^3x$	0

Step 5: $\Lambda(x) = 1 + \alpha^3x$, indicating a single error at $X_1 = \alpha^3$.

Step 6: The error magnitude polynomial is

$$\begin{aligned}\Omega(x) &= \Lambda(x)(1 + S(x)) = (1 + \alpha^3x)(1 + \alpha^2x + \alpha^2x^2 + \alpha^6x^3 + x^4) \bmod x^5 \\ &= (1 + (\alpha^2 + \alpha^3)x + (\alpha^2 + \alpha^5)x^2 + (\alpha^6 + \alpha^5)x^3 \\ &\quad + (1 + \alpha^9)x^4 + \alpha^3x^5) \bmod x^5 \\ &= (1 + \alpha^5x + \alpha^3x^2 + \alpha x^3 + \alpha^6x^4 + \alpha^3x^5) \bmod x^5 \\ &= 1 + \alpha^5x + \alpha^3x^2 + \alpha x^3 + \alpha^6x^4\end{aligned}$$

The error/erasure locator polynomial

$$\begin{aligned}\Psi(x) &= \Lambda(x)\Gamma(x) = (1 + \alpha^3x)(1 + \alpha^5x) \\ &= (1 + \alpha^2x + \alpha x^2)\end{aligned}$$

The error magnitude

$$\begin{aligned}e_k &= \frac{-X_k \Omega(X_k^{-1})}{\Psi'(X_k^{-1})} = \frac{-X_k [1 + \alpha^5 X_k^{-1} + \alpha^3 X_k^{-2} + \alpha X_k^{-3} + \alpha^6 X_k^{-4}]}{\alpha^2} = \alpha^3 \\ e_3 &= \alpha^3\end{aligned}$$

and erasure magnitude

$$f_k = \frac{-Y_k \Omega(Y_k^{-1})}{\Psi'(Y_k^{-1})}$$

$$f_5 = \alpha$$

The corrected code word is

$$\begin{aligned} c(x) &= r(x) + e(x) + f(x) \\ &= (\alpha^3 x^2 + x^4 + x^6) + \alpha^3 x^3 + \alpha x^5 \end{aligned}$$

4.6 Performance Analysis of RS Codes

A RS (n, k) code with minimum distance $d_{\min} = n - k + 1$ is able to correct $t_{ec} = (n - k)/2$ symbol errors. The bit error probability for RS codes using hard decision decoding is often approximated by [5].

$$P_b \approx \frac{1}{n} \sum_{i=t_{ec}+1}^t i \binom{n}{i} P^i (1-P)^{n-i} \quad (4.48)$$

4.6.1 BER Performance of RS Codes for BPSK Modulation in AWGN and Rayleigh Fading Channels

The redundancy introduced by RS code increases the channel-symbol transmission rate, reducing the received $\frac{E_b}{N_0}$. For a code with rate R , for BPSK in AWGN channel and Rayleigh fading channel, Eqs. (2.3) and (2.6) become

$$\begin{aligned} P &= Q\left(\sqrt{2R \frac{E_b}{N_0}}\right) \\ P &= \frac{1}{2}\left(1 - \sqrt{\frac{R\bar{\gamma}}{1+R\bar{\gamma}}}\right) \quad (4.49) \end{aligned}$$

where $R = k/n$.

The following MATLAB program is used to compare the theoretical decoding error probability of different RS codes with BPSK modulation in AWGN channel.

Program 4.1 Program to compare the decoding error probability of different RS codes

```

clearall;clc;
Eb_N0_dB = [0:0.3:15]; % multiple Eb/N0 values
EbN0Lin = 10.^ (Eb_N0_dB/10);
% Uncoded
P_b0 = 0.5* erfc( sqrt( EbN0Lin ) );
% RS (127, 106), t=10
m=7;
n6 = 2^m -1;
k6 = 106;
t6 = 10;
R_c6 = k6/n6;
p = 0.5* erfc( sqrt(R_c6 * EbN0Lin) );
sum = 0
for j=t6+1:n6
sum = sum +j*factorial(n6)/factorial(j)/factorial (n6-j) .*p.^j .* (1-p).^(n6-j)
end
P_b1 = sum/n6;
% RS (127, 106), t=10
m=5;
n6 = 2^m -1;
k6 = 15;
t6 = 8;;
R_c6 = k6/n6;
p = 0.5* erfc( sqrt(R_c6 * EbN0Lin) );
sum = 0
for j=t6+1:n6
sum = sum +j*factorial(n6)/factorial(j)/factorial (n6-j) .*p.^j .* (1-p).^(n6-j)
end
P_b2 = sum/n6;
m=5;
n6 = 2^m -1;
k6 = 21;
t6 = 5;
R_c6 = k6/n6;
p = 0.5* erfc( sqrt(R_c6 * EbN0Lin) );
sum = 0
for j=t6+1:n6
sum = sum +j*factorial(n6)/factorial(j)/factorial (n6-j) .*p.^j .* (1-p).^(n6-j)
end
P_b3 = sum/n6;
m=5;%7;

```

```

n6 = 2^m -1;
k6 = 27;
t6 = 2;
R_c6 = k6/n6;
p = 0.5* erfc( sqrt(R_c6 * EbN0Lin) );
sum = 0
for j=t6+1:n6
sum = sum +j*factorial(n6)/factorial(j)/factorial (n6-j) .*p.^j .* (1-p).^(n6-j)
end
P_b4 = sum/n6;
figure
semilogy(Eb_N0_dB, P_b0, '-',Eb_N0_dB,P_b1, '-x', Eb_N0_dB, P_b2, '-
*',Eb_N0_dB, P_b3, '-+', Eb_N0_dB, P_b4, '-d');
xlabel('Eb/N0 (dB)'), ylabel('BER');
legend('Uncoded','RS (127,106)', 'RS (31,15)', 'RS (31,21)', 'RS (31,27)');
axis([0 15 1e-6 1e-0]);

```

The decoding error probability obtained from the above program for RS (127,106) and for RS code of length 31 with different dimensions k is shown in Fig. 4.11.

From Fig. 4.11, it can be observed that the decoder error probability approach is increasingly lower as the E_b/N_0 and code dimension decrease. This can be attributed to the highly imperfect nature of RS codes.

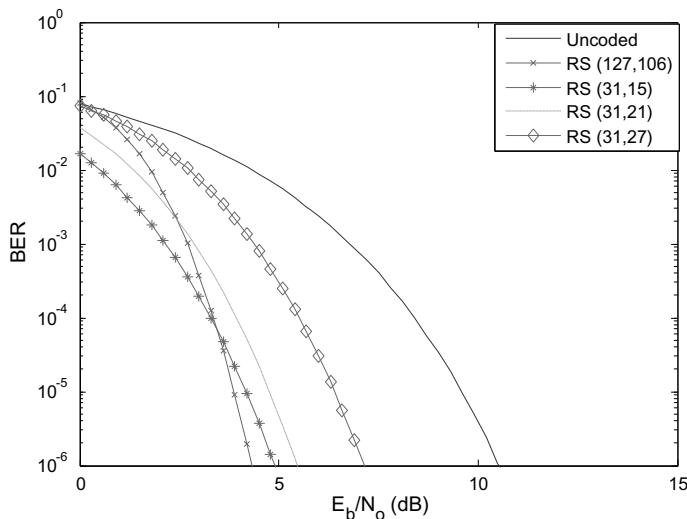


Fig. 4.11 Decoding error probability for RS codes using coherent BPSK over an AWGN channel

The following MATLAB program compares the theoretical BER performance of (127, 63) RS code with BPSK modulation in AWGN and Rayleigh fading channels.

Program 4.2 Program to compare the decoding error probability of an RS code in AWGN and Rayleigh fading channels using coherent BPSK modulation.

```

clearall;clc;
Eb_N0_dB = [0:0.3:15]; % multiple Eb/N0 values
EbN0Lin = 10.^ (Eb_N0_dB/10);
% Uncoded
P_bawgn = 0.5* erfc( sqrt( EbN0Lin) );
p_bRay = 1/2 - 1/2*(1+1./EbN0Lin).^( -1/2);
% RS (127, 106), t=10
m=7; n6 = 2^m -1;
k6 = 63;
t6 = 32;
R_c6 = k6/n6;
p = 0.5* erfc( sqrt(R_c6 * EbN0Lin) );
p1 = 1/2 - 1/2*(1+1./(R_c6 * EbN0Lin)).^( -1/2);
sum = 0
for j=t6+1:n6
sum = sum +j*factorial(n6)/factorial(j)/factorial (n6-j) .*p.^j .* (1-p).^ (n6-j)
end
P_b1 = sum/n6;
sum = 0
for j=t6+1:n6
sum = sum +j*factorial(n6)/factorial(j)/factorial (n6-j) .*p1.^j .* (1-
p1).^(n6-j)
end
P_b2 = sum/n6;
figure
semilogy(Eb_N0_dB, p_bRay, '- ', Eb_N0_dB, P_bawgn, '-- ', Eb_N0_dB, P_b1, '-
*', Eb_N0_dB, P_b2, '-d');
xlabel('E_b/N_o (dB)'), ylabel('BER');
legend('Uncoded Rayleigh ','Uncoded AWGN','RS (127,63) AWGN', 'RS
(127,63) Rayleigh');
axis([0 15 1e-6 1e-0]);

```

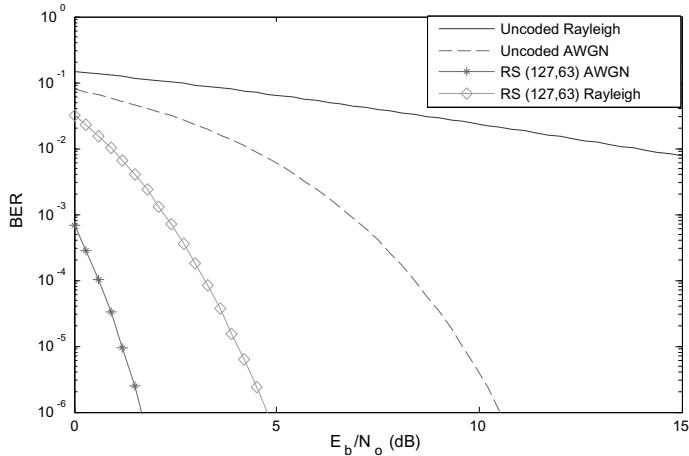


Fig. 4.12 Decoding error probability for (127,63) RS codes using coherent BPSK over an AWGN channel and Rayleigh fading channel

The decoding error probability obtained from the above program for RS (127,63) is shown in Fig. 4.12.

From Fig. 4.12, it is seen that the coded AWGN and Rayleigh fading channels exhibit much better BER performance than the uncoded AWGN and Rayleigh fading channels.

4.6.2 BER Performance of RS Codes for Non-coherent BFSK Modulation in AWGN and Rayleigh Fading Channels

From Eq. (2.25), for BFSK ($M = 2$), the probability of bit error P for AWGN and Rayleigh fading channels can be expressed as

$$P = \frac{1}{2} \exp\left[-\frac{RE_b}{2N_0}\right] \text{ AWGN}$$

$$P = \frac{1}{2 + R\bar{\gamma}} \text{ Rayleigh fading} \quad (4.50)$$

Program 4.3 Program to compare the decoding error probability of an RS code in AWGN and Rayleigh fading channels using non-coherent BFSK modulation.

```

clearall;clc;
Eb_N0_dB = [0:0.3:15];
K=5; m= 7;
EbN0Lin = 10.^ (Eb_N0_dB/10);
P_bawgn = 0.5*exp(-0.5*EbN0Lin); % theoretical ber
p_bRay = 1./ (EbN0Lin+2);
n6 = 2^m -1;
k6 = 63;
t6 = 32;
R_c6 = k6/n6;
p = 0.5*exp(-0.5*R_c6*EbN0Lin);
p1 = 1./ (2+R_c6 * EbN0Lin);
sum = 0
for j=t6+1:n6
sum = sum +j*factorial(n6)/factorial(j)/factorial (n6-j) .*p.^j .* (1-p).^(n6-j)
end
P_b1 = sum/n6;
sum = 0
for j=t6+1:n6
sum = sum +j*factorial(n6)/factorial(j)/factorial (n6-j) .*p1.^j .* (1-
p1).^(n6-j)
end
P_b2 = sum/n6;
figure
semilogy(Eb_N0_dB, p_bRay, '-',Eb_N0_dB,P_bawgn, '--', Eb_N0_dB, P_b1, '-*
',Eb_N0_dB, P_b2, '-d');
xlabel('E_b/N_o (dB)'), ylabel('BER');
legend('Uncoded Rayleigh ','Uncoded AWGN','RS (127,63) AWGN', 'RS
(127,63) Rayleigh');
axis([0 15 1e-6 1e-0]);

```

The decoding error probability obtained from the above program for RS (127,63) is shown in Fig. 4.13.

From Fig. 4.13, it is seen that the coded AWGN and Rayleigh fading channels exhibit much better BER performance than the uncoded AWGN and Rayleigh fading channels. However, the performance is not better as compared to that of BPSK modulation.

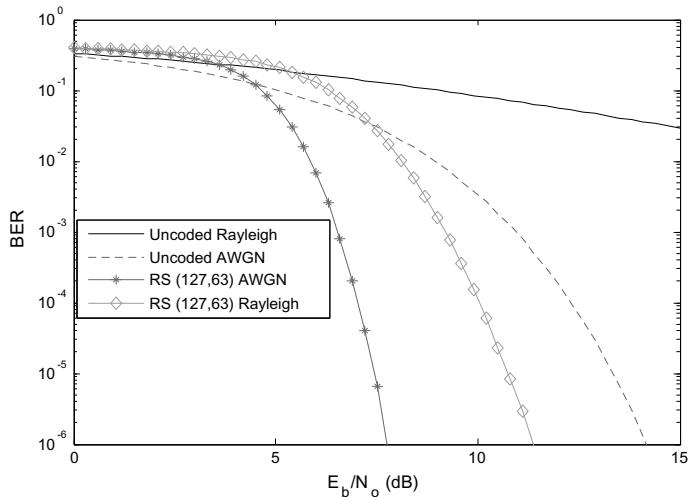


Fig. 4.13 Decoding error probability for (127,63) RS codes using non-coherent BPSK over an AWGN channel and Rayleigh fading channel

4.7 Problems

- 1 Construct encoder circuit using shift register for (15,7) cyclic code generated by $g(x) = 1 + x^4 + x^6 + x^7 + x^8$. and find the code word corresponding to the information sequence (1001011).
- 2 Construct a shift register decoder for the(15,11) cyclic Hamming code generated by $g(x) = 1 + x + x^4$ and decode the received word $r = (111100000100100)$.
- 3 Let C_1 be the binary CRC code of length 13 generated by $g(x) = 1 + x + x^3$. Compute the code polynomial in C_1 and the associated code word for the message polynomial $m(x) = 1 + x^2 + x^5 + x^7 + x^9$
- 4 Design a four-error-correcting binary BCH code of length 15.
- 5 Let the transmission code be the triple-error-correcting binary BCH code of length 31. The generator polynomial is $g(x) = 1 + x + x^2 + x^3 + x^5 + x^7 + x^8 + x^9 + x^{10} + x^{11} + x^{15}$. Use Berlekamp's algorithm to decode the following received vector $r = (0100000000001000000000000100000)$.
- 6 Let the transmission code be the double-error-correcting binary BCH code of length 15. The generator polynomial is $g(x) = 1 + x^4 + x^6 + x^7 + x^8$. Use Berlekamp's algorithm to decode the following received vector $r = (00f0000000000000)$. The f indicates erasure.
- 7 Construct a generator polynomial for a double-error-correcting Reed–Solomon code of length 7 and determine the number of code words it does have.
- 8 Determine the weight distribution for the RS code of problem 1.
- 9 Compute a generator polynomial for a triple-error-correcting Reed–Solomon code of length 31.

- 10 Construct a generator polynomial for a (63,57) RS code and determine the code words. It does have.
- 11 Let the transmission code be the double-error-correcting RS code of length 7. Use the Berlekamp–Massey algorithm to decode the following received vector $r = (00010\alpha 0)$.
- 12 Let the transmission code be the double-error-correcting RS code of length 7. Use the Berlekamp–Massey algorithm to decode the following received vector $r = (1010000)$.
- 13 Let the transmission code be the double-error-correcting RS code of length 7. Use the Berlekamp–Massey algorithm to decode the following received vector $r = (\alpha^3\alpha 1\alpha\alpha^2 00)$.
- 14 Let the transmission code be the triple-error-correcting RS code of length 15. Decode the received vector $r = (000\alpha^7 000000\alpha^{11} 0000)$ using Berlekamp–Massey algorithm.
- 15 Let the transmission code be the double-error-correcting RS code of length 15. Use the Berlekamp–Massey algorithm to decode the following received vector $r = (1001000000000000)$.
- 16 Let the transmission code be the triple-error-correcting RS code of length 31. Decode the following received vector $r = (\alpha^2 000000000000\alpha^{21} 0000000\alpha^7 000000000)$ using the Berlekamp–Massey algorithm.
- 17 Let the transmission code be the double-error-correcting RS code of length 7. Use the Berlekamp–Massey algorithm to decode the following received vector $r = (00\alpha^3 f 101)$.

4.8 MATLAB Exercises

1. Write a MATLAB program to simulate the performance BPSK modulation in AWGN and Rayleigh fading channels and compare with theoretical results shown in Chap. 2.
2. Write a MATLAB program to simulate the performance of RS coded SFH-CDMA using BFSK modulation and compare with the uncoded theoretical results shown in Chap. 2.
3. Write a MATLAB program to simulate the BER performance of an RS code in AWGN and Rayleigh fading channels using BPSK modulation and compare with the theoretical results shown in Fig. 4.4.
4. Write a MATLAB program to simulate the BER performance of an RS code in AWGN and Rayleigh fading channels using BFSK modulation and compare with the theoretical results shown in Fig. 4.5.
5. Write a MATLAB program to simulate the BER performance of an RS code in AWGN and Rayleigh fading channels using MFSK modulation for $M > 2$.

References

1. G.D. Forney, On decoding BCH codes. IEEE Trans. Info. Theory **IT-11**, 549–557 (1965, October)
2. J.L. Massey, Shift register synthesis and BCH decoding. IEEE Trans. Info. Theory **IT-15**(1), 122–127 (1969, January)
3. E.R. Berlekamp, *Algebraic Coding Theory*, rev. ed. (Aegean Park Press, Laguna Hills, 1984)
4. R.T. Chien Cyclic decoding procedure for the Bose-Chaudhuri-Hocquenghem Codes. " IEEE Trans. Info. Theory, **IT-10**(1) 357–363 (1964, October)
5. K.-L. Du, M.N.S. Swamy, *Wireless Communications: Communication Systems from RF Subsystems to 4G Enabling Technologies* (Cambridge University Press, 2010)

Chapter 5

Convolutional Codes



In the convolutional coding, the message bits come in serially instead of large blocks. The name convolutional codes are due to the fact that the redundant bits are generated by the use of modulo-2 convolutions in a convolutional encoder. The convolutional encoder can be considered as finite-state machine consisting of an M-stage shift register, modulo-2 adders and multiplexers. The rate of a convolutional encoder with k inputs and n outputs is k/n . Often the manufacturers of convolutional code chips specify the code by parameters (n, k, L) . The quantity L is called the constraint length of the code that represents the maximum number of bits in a single output stream that can be affected by any input bit.

5.1 Structure of Non-systematic Convolutional Encoder

Consider a rate 1/3 convolutional encoder as shown in Fig. 5.1. The binary data stream $x(n) = (x(0), x(1), x(2), \dots)$ is fed into shift register containing a series of memory elements. The contents of the memory elements are tapped and added according to modulo-2 addition to create the coded output data streams.

$$\begin{aligned}y_1(n) &= (y_1(0), y_1(1), y_1(2), \dots), \\y_2(n) &= (y_2(0), y_2(1), y_2(2), \dots) \text{ and} \\y_3(n) &= (y_3(0), y_3(1), y_3(2), \dots).\end{aligned}$$

Then, these output coded data streams are multiplexed to create a single-coded output data stream.

$$Y = (y_1(0), y_2(0), y_3(0), y_1(1), y_2(1), y_3(1), y_1(2), y_2(2), y_3(2), \dots)$$

The output stream $y_1(n)$, $y_2(n)$ and $y_3(n)$ can be represented as.

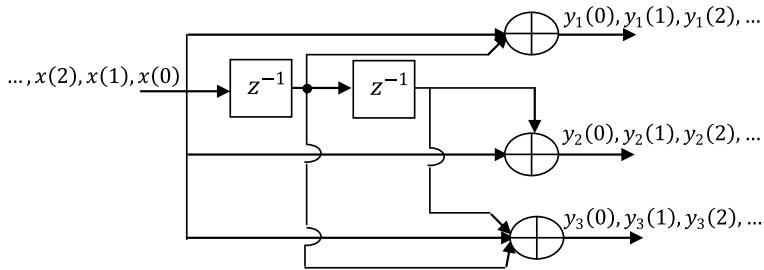


Fig. 5.1 A rate 1/3 linear convolutional encoder

$$y_1(n) = x(n) + x(n - 1)$$

$$y_2(n) = x(n) + x(n - 2)$$

$$y_3(n) = x(n) + x(n - 1) + x(n - 2)$$

Example 5.1 Prove the encoder shown in Fig. 5.1 is linear convolutional encoder.

Proof Let the input $x_1(n) = (11101)$. Then, the corresponding coded output sequences.

$$y_1(n) = (1001110)$$

$$y_2(n) = (1101001)$$

$$y_3(n) = (1010011)$$

The convolutional code word corresponding to $x_1(n) = (11101)$ is then.

$$Y_1 = (111, 010, 001, 110, 100, 101, 011)$$

Let the input $x_2(n) = (10010)$.

The corresponding coded output sequences are.

$$y_1(n) = (1101100)$$

$$y_2(n) = (1011010)$$

$$y_3(n) = (1111110)$$

The convolutional code word corresponding to $x_2(n) = (10010)$ is.

$$Y_2 = (111, 101, 011, 111, 101, 011, 000)$$

Let the input $x(n) = x_1(n) + x_2(n) = (01111)$.

The corresponding coded output sequences.

$$y_1(n) = (0100010)$$

$$y_2(n) = (0100010)$$

$$y_3(n) = (0101101)$$

The convolution code word corresponding to $x(n) = (01111)$ is given as follows:

$$Y = (000, 111, 010, 001, 001, 110, 011)$$

$$\begin{aligned} Y_1 + Y_2 &= (111, 010, 001, 110, 100, 101, 011) + (111, 101, 011, 111, 101, 011, 000) \\ &= (000, 111, 010, 001, 001, 110, 011) \\ &= Y \end{aligned}$$

“A convolutional encoder is linear, if Y_1 and Y_2 are the code words corresponding to inputs $x_1(n)$ and $x_2(n)$, respectively, then $(Y_1 + Y_2)$ is the code word corresponding to the input $x_1(n) + x_2(n)$.” Hence, the convolutional encoder in the problem is proved to be linear.

5.1.1 Impulse Response of Convolutional Codes

The impulse response stream $g_i(n)$ for the input $x(n) = (1000 \dots)$ for the encoder shown in Fig. 5.1 can be represented as follows:

The impulse response $g_1(n)$ can be represented by

$$g_1(n) = x(n) + x(n - 1)$$

The impulse response $g_2(n)$ can be represented by

$$g_2(n) = x(n) + x(n - 2)$$

The impulse response $g_3(n)$ can be represented by

$$g_3(n) = x(n) + x(n - 1) + x(n - 2)$$

Thus, the impulse responses for the encoder are

$$g_1(n) = (110)$$

$$g_2(n) = (101)$$

$$g_3(n) = (111)$$

Since there are two memory elements in the shift register of the encoder, each bit in the input data stream can affect at most 3 bits; hence, the length of the above impulse response sequence is 3.

Since the convolutional encoder can be described by discrete convolutional operation, if the information sequence $x(n)$ is input to the encoder, the three outputs are given by

$$\begin{aligned}y_1(n) &= x(n) * g_1(n) \\y_2(n) &= x(n) * g_2(n) \\y_3(n) &= x(n) * g_3(n)\end{aligned}$$

where $*$ represents the convolution operation. In the D-transform domain, the three outputs can be represented as.

$$\begin{aligned}Y_1(D) &= X(D)G_1(D) \\Y_2(D) &= X(D)G_2(D) \\Y_3(D) &= X(D)G_3(D)\end{aligned}$$

The D denotes the unit delay introduced by the memory element in the shift register. The use of D -transform is most common in the coding literature. The delay operator D is equivalent to the indeterminate z^{-1} of the z -transform. The D -transforms of the impulse responses of the above encoder are

$$\begin{aligned}G_1(D) &= 1 + D \\G_2(D) &= 1 + D^2 \\G_3(D) &= 1 + D + D^2\end{aligned}$$

Hence, the encoder shown in Fig. 5.1 can be described by a generator matrix.

$$G(D) = [G_1(D) \ G_2(D) \ G_3(D)]$$

The transform of the encoder output can be expressed as

$$Y(D) = X(D)G(D)$$

where

$$Y(D) = [Y_1(D) \ Y_2(D) \ Y_3(D)]$$

The $G(D)$ is called the transfer function matrix of the encoder shown in Fig. 5.1.

Example 5.2 Determine the output codeword of the encoder shown in Fig. 5.1 using the transfer function matrix if the input sequence $X = (11101)$.

Solution The D -transform of the input sequence x is given by

$$X(D) = 1 + D + D^2 + D^4$$

The D -transform of the encoder output follows as

$$\begin{aligned} Y(D) &= [1 + D + D^2 + D^4][1 + D^3 + D^2 + D + D^2] \\ &= [1 + D^3 + D^4 + D^5 + D^6][1 + D^2 + D^5 + D^6] \end{aligned}$$

Inverting the D -transform we get.

$$\begin{aligned} y_1(n) &= (1001110) \\ y_2(n) &= (1101001) \\ y_3(n) &= (1010011) \end{aligned}$$

Then, the output codeword y is

$$y = (111, 010, 001, 110, 100, 101, 011)$$

5.1.2 Constraint Length

The constraint length “ L ” of a convolutional code is the length of longest input shift register with maximum number of memory elements plus one.

5.1.3 Convolutional Encoding Using MATLAB

The following MATLAB program illustrates the computation of the output codeword of the encoder shown in Fig. 5.1 for input sequence $x_2 = (10010)$.

Program 5.1 MATLAB program to determine the output codeword of the encoder shown in Fig. 5.1.

```
clear all;clc;
x=[1 0 0 1 0];
y1 = mod(conv(x,[1 1 0]),2);
y2 = mod(conv(x,[1 0 1]),2);
y3 = mod(conv(x,[1 1 1]),2);
Y123 = [y1;y2;y3];
```

The above program outputs the following codeword.

$$Y = [1 1 1 1 0 1 0 1 1 1 1 1 0 1 0 1 1 0 0 0]$$

5.2 Structure of Systematic Convolutional Encoder

A convolutional code in which the input data appears as a part of the code sequence is said to be systematic. A rate 1/3 systematic convolutional encoder is shown in Fig. 5.2.

5.3 The Structural Properties of Convolutional Codes

5.3.1 State Diagram

The contents of memory elements of a convolutional encoder provides mapping between the input bits and the output bits. An encoder with j memory elements can assume any one of 2^j possible states. The encoder can only move between states. Each branch in the state diagram has a label of the form $X/YYY \dots$, where X is the input bit that causes the state transition and $YYY \dots$ is the corresponding output bits. The encoder shown in Fig. 5.1 consists of two memory elements, and hence, the two binary elements can assume any one of four states designated by $S_0 - 00; S_1 - 10; S_2 - 01; S_3 - 11$.

For the encoder shown in Fig. 5.1, the state diagram is shown in Fig. 5.3.

5.3.2 Catastrophic Convolutional Codes

A convolutional code is said to be catastrophic if its encoder generates all-zero output sequence for a nonzero input sequence. A catastrophic code can cause an unlimited number of data errors for a small number of errors in the received code word. The following Theorem [1] can be used to verify whether a convolutional code is catastrophic.

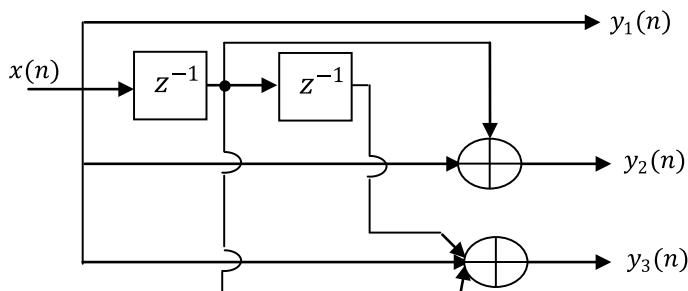
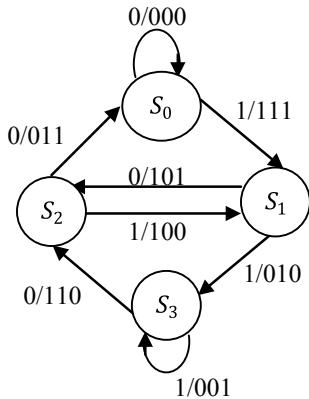


Fig. 5.2 A rate 1/3 systematic convolutional encoder

Fig. 5.3 State diagram of non-systematic convolutional encoder shown in Fig. 5.1



Theorem 5.1 A rate $1/n$ convolutional code with transfer function matrix $G(D)$ with generated sequences having the transforms $\{G_0(D), G_1(D), \dots, G_{n-1}(D)\}$ is not catastrophic if and only if

$$\text{GCD}(G_0(D), G_1(D), \dots, G_{n-1}(D)) = D^l$$

for some non-negative integer l .

Example 5.3 Determine the encoder shown in Fig. 5.4 generates a catastrophic convolutional code or not.

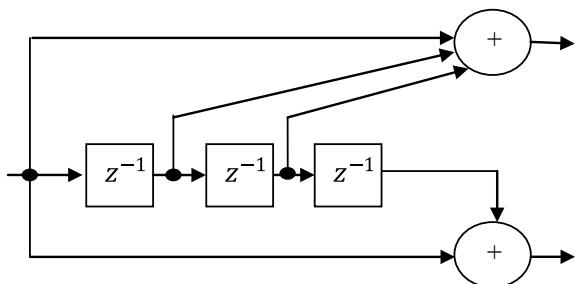
Solution From the encoder diagram shown in Fig. 5.4, the impulse responses are $g_1 = (1110)$ and $g_2 = (1001)$.

The transform of the generator sequence $g_1 = (1110)$ is $G_1(D) = 1 + D + D^2$.

The transform of the generator sequence $g_2 = (1001)$ is $G_2(D) = 1 + D^3$.

$$\begin{aligned} \text{GCD}[G_1(D), G_2(D)] &= \text{GCD}[1 + D + D^2, 1 + D^3] \\ &= 1 + D + D^2 \\ &\neq D^l \end{aligned}$$

Fig. 5.4 A rate-1/2 convolutional encoder



Thus, the code is catastrophic for any integer l , where GCD stands for greatest common divisor.

5.3.3 Transfer Function of a Convolutional Encoder

The signal-flow graph for a convolutional encoder can be obtained by splitting the state S_0 into a source node and sink node by modifying the labels of the branches. For a given branch, we label $Y^i X^j$ where j is the weight of the input vector X and i is the weight of the output vector Y (the number of nonzero coordinates).

Example 5.4 Determine the transfer function of the systematic convolutional encoder shown in Fig. 5.2.

Solution The state diagram of the systematic convolutional encoder is shown in Fig. 5.5.

The signal-flow graph of the above state diagram is shown in Fig. 5.6. In this signal-flow graph, the self loop at node S_0 is eliminated as it contributes nothing to the distance properties of a code relative to the all-zero code sequence. Now, by using the signal-flow graph reduction techniques and Mason's formula, the transfer function can be obtained.

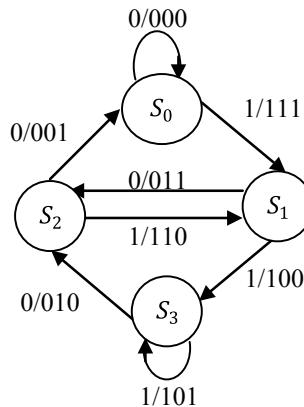


Fig. 5.5 State diagram of systematic convolutional encoder shown in Fig. 5.2

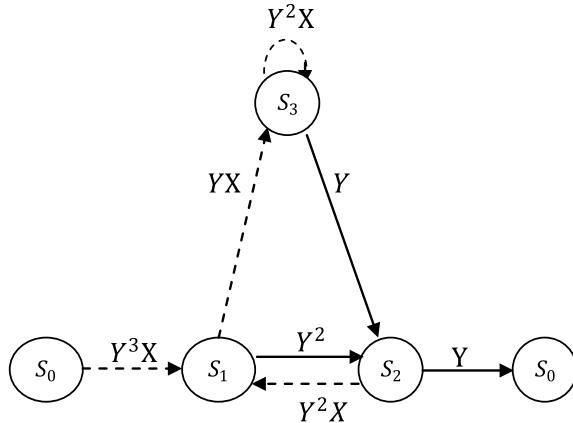
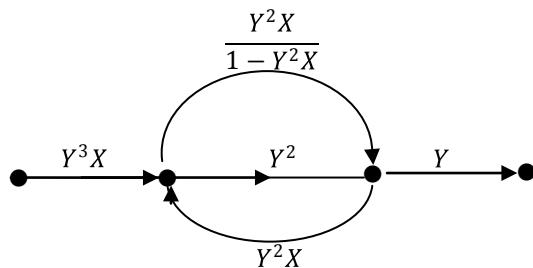
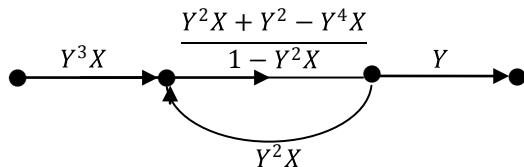


Fig. 5.6 Signal-flow graph of the above state diagram is shown in Fig. 5.5

By using reduction techniques, the above signal-flow graph can be simplified as

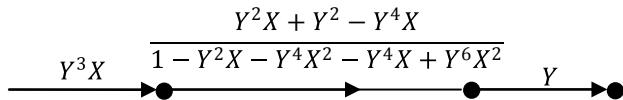


Further, the parallel branches with gains Y^2 and $\frac{Y^2X}{1-Y^2X}$ can be combined as a single branch with gain $Y^2 + \frac{Y^2X}{1-Y^2X} = \frac{Y^2X+Y^2-Y^4X}{1-Y^2X}$ as follows



Further, the loop can be replaced by a branch with gain

$$\frac{\frac{Y^2X+Y^2-Y^4X}{1-Y^2X}}{1 - Y^2X \frac{Y^2X+Y^2-Y^4X}{1-Y^2X}} = \frac{Y^2X + Y^2 - Y^4X}{1 - Y^2X - Y^4X^2 - Y^4X + Y^6X^2}$$



Thus, the transfer function is given by

$$\begin{aligned} T(Y) &= Y^3X \frac{Y^2X + Y^2 - Y^4X}{1 - Y^2X - Y^4X^2 - Y^4X + Y^6X^2} Y \\ &= \frac{Y^6X^2 + Y^6X - Y^8X^2}{1 - Y^2X - Y^4X^2 - Y^4X + Y^6X^2} \end{aligned}$$

Example 5.5 Consider the following non-systematic convolutional encoder and determine its transfer function (Fig. 5.7).

Solution The state diagram of the non-systematic convolutional encoder is shown in Fig. 5.8.

The signal-flow graph of the above state diagram is shown in Fig. 5.9.

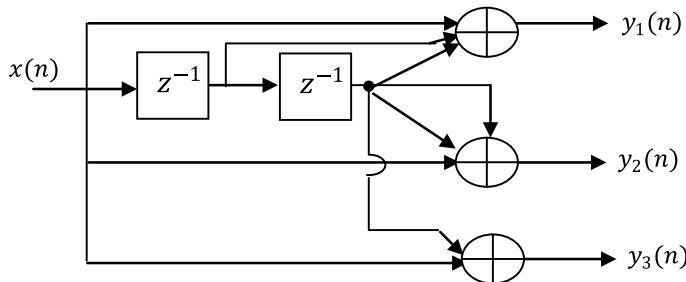
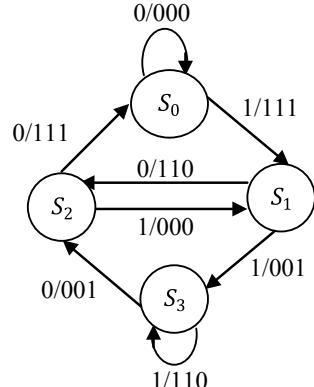


Fig. 5.7 Encoder for a rate-1/3 convolutional code

Fig. 5.8 State diagram of non-systematic convolutional encoder shown in Fig. 5.7



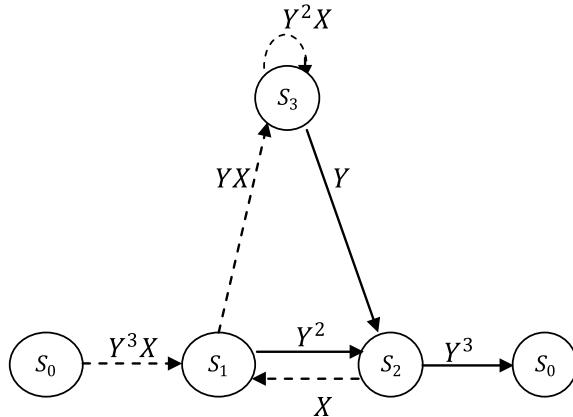
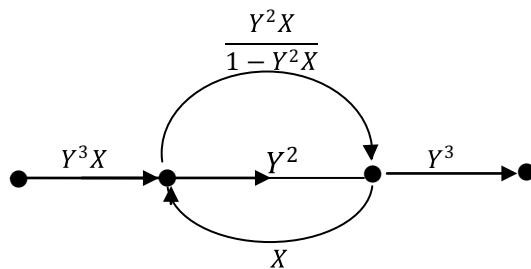
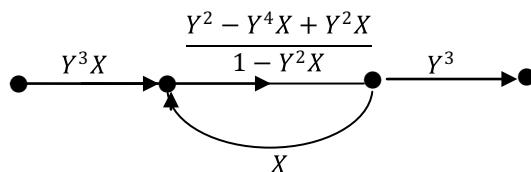


Fig. 5.9 Signal-flow graph of the state diagram shown in Fig. 5.8

By using reduction techniques, the above signal-flow graph can be simplified as follows:



Further, the parallel branches with gains Y^2 and $\frac{Y^2X}{1-Y^2X}$ can be combined as a single branch with gain $Y^2 + \frac{Y^2X}{1-Y^2X} = \frac{Y^2 - Y^4X + Y^2X}{1 - Y^2X}$ as follows

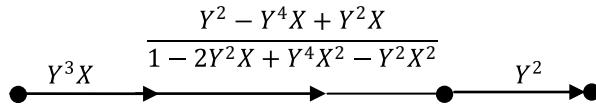


Further, the loop can be replaced by a branch with gain.

$$\frac{\frac{Y^2 - Y^4X + Y^2X}{1 - Y^2X}}{1 - X \frac{Y^2 - Y^4X + Y^2X}{1 - Y^2X}} = \frac{Y^2 - Y^4X + Y^2X}{1 - 2Y^2X + Y^4X^2 - Y^2X^2}$$

Thus, the transfer function is given by

$$\begin{aligned} T(Y, X) &= Y^3 X \frac{Y^2 - Y^4 X + Y^2 X}{1 - 2Y^2 X + Y^4 X^2 - Y^2 X^2} Y^3 \\ &= \frac{Y^8 X - Y^{10} X^2 + Y^8 X^2}{1 - 2Y^2 X + Y^4 X^2 - Y^2 X^2} \end{aligned}$$



5.3.4 Distance Properties of Convolutional Codes

An upper bound on the minimum free distance of a rate $1/n$ convolutional code is given by [2].

$$d_f \leq \max_{l>1} \lfloor \frac{2^{l-1}}{2^l - 1} (L + l - 1)n \rfloor \quad (5.1)$$

where $\lfloor x \rfloor$ denotes the largest integer contained in x .

The transfer function also yields the distance properties of the code. The minimum distance of the code is called the minimum free distance denoted by d_f . The d_f is the lowest power in the transfer function.

In Example 5.4, since the lowest power in the transfer function is 6, the d_f for the systematic convolutional encoder considered in this example is 6, whereas in Example 5.5, the lowest power in the transfer function is 8. Hence, the minimum free distance for the non-systematic encoder considered in this example is 8.

From the above two examples, it is observed that the minimum free distance for non-recursive systematic convolutional code is less than that of a non-recursive non-systematic convolutional codes of the same rate and constraint length. The bounds on the minimum free distance for various codes are developed in [3, 4]. The bounds on the free distance for various systematic and non-systematic codes of the same rate and constraint length are tabulated in Table 5.1.

5.3.5 Trellis Diagram

The state diagram does not contain time information required in decoding. Hence, trellis diagram is developed to overcome the disadvantage. The trellis diagram is an

Table 5.1 Bounds on the free distance for various systematic and non-systematic codes of the same rate and constraint length

Rate	Constraint length	Systematic codes maximum free distance	Non-systematic codes maximum free distance
1/3	2	5	5
	3	6	8
	4	8	10
	5	9	12
1/2	2	3	3
	3	4	5
	4	4	6
	5	5	7

expansion of state diagram by adding a time axis for time information. In the trellis diagram, the nodes are arranged vertically representing the states of the encoder and each node corresponding to a state of the encoder after a transition from the previous node for an input bit, the horizontal axis represents time and the labels on the branches represent the encoder output bits for a state transition and the input bit causing the transition.

For a (n, k) convolutional code with memory order m , there are 2^m nodes at each time increment t and there are 2^k branches leaving each node for $t \leq m$. For $t > m$, there are also 2^k branches entering the node.

For an encoder with single input sequence of B bits, the trellis diagram must have $B + m$ stages with the first and last stages starting, and stopping, respectively, in state S_0 . Thus, there are 2^B distinct paths through trellis each corresponding to the code word of the length $(B + m)$.

Example 5.6 The impulse responses of a convolutional encoder are given by $g_1 = [101]$; $g_2 = [111]$.

- Draw the encoder.
- Draw the state diagram.
- Draw the trellis diagram for the first three stages.

Solution

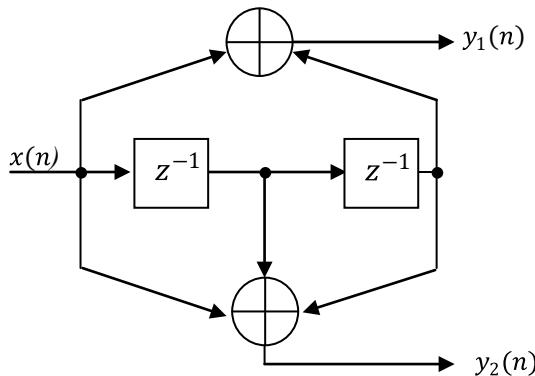
- From the impulse responses $g_1 = [101]$; $g_2 = [111]$, the output stream $y_1(n)$ can be represented as follows

$$y_1(n) = x(n) + x(n - 2)$$

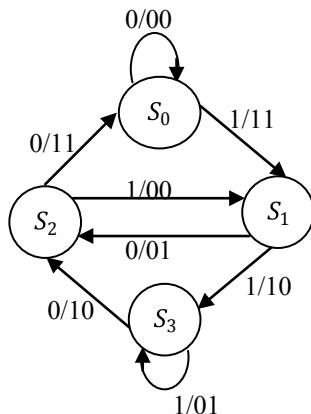
The output stream $y_2(n)$ can be represented as follows

$$y_2(n) = x(n) + x(n - 1) + x(n - 2)$$

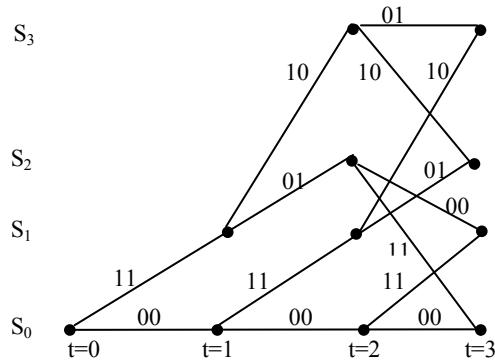
Hence, the corresponding encoder is as follows



- ii. This rate-1/2 encoder has two memory cells. So, the associated state diagram has four states as shown below.

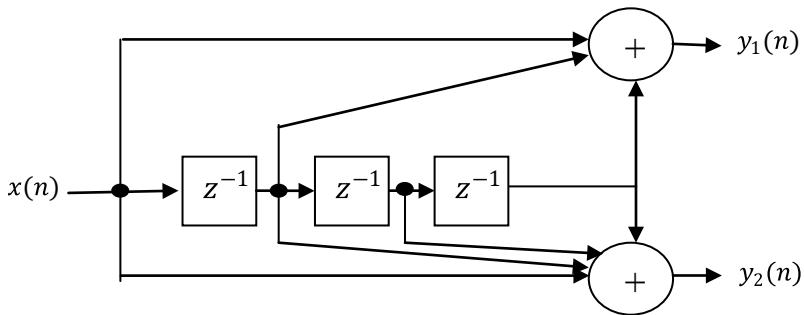


- iii. The trellis diagram is an extension of the state diagram that explicitly shows the passage of time. The first three stages of the trellis diagram corresponding to the encoder are as follows:

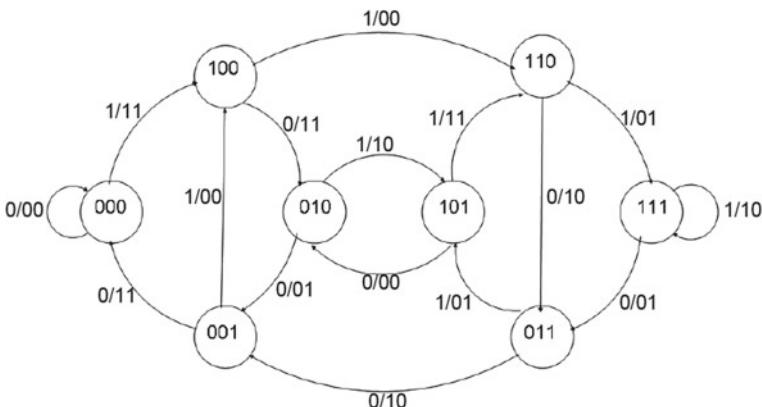


Example 5.7 Consider the following convolutional encoder

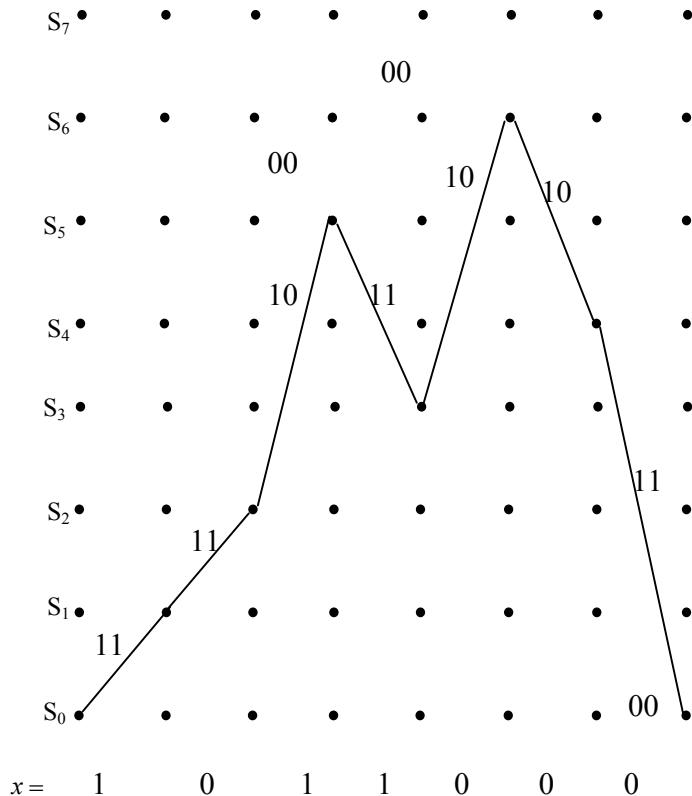
- Draw the state diagram.
- Draw the trellis diagram and find the code word for the input $x = (1\ 0\ 1\ 1)$.



Solution The three binary memory elements can assume any one of four states $S_{\{0\}}—000; S_{\{1\}}—100; S_{\{2\}}—010; S_{\{3\}}—110; S_{\{4\}}—001; S_{\{5\}}—101; S_{\{6\}}—011; S_{\{7\}}—111$. The state diagram is as follows



For the input sequence $u = [1011]$ starting from all-zeros state goes through the states $S = (000; 100; 010; 101; 110)$. Then, by feeding two zero inputs, it goes through states $S = (011; 001; 000)$ resulting in the output codeword $c = (11, 11, 10, 11, 10, 11)$.



The codeword obtained from the above trellis is $c = (11, 11, 10, 11, 10, 10, 11)$.

Example 5.8 The impulse responses of a convolutional encoder are given by

$$g_1 = [111]; \quad g_2 = [111]; \quad g_3 = [110]$$

- Draw the encoder.
- Draw the state diagram.
- Draw the trellis diagram for the length-3 input sequence.

Solution

- i. From the impulse responses $g_1 = [111]$; $g_2 = [111]$; $g_3 = [110]$, the output stream $y_1(n)$ can be represented as follows

$$y_1(n) = x(n) + x(n - 1) + x(n - 2)$$

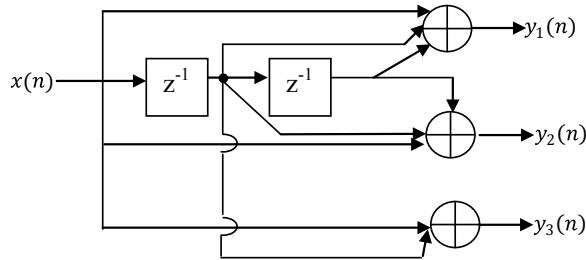
The output stream $y_2(n)$ can be represented as follows

$$y_2(n) = x(n) + x(n - 1) + x(n - 2)$$

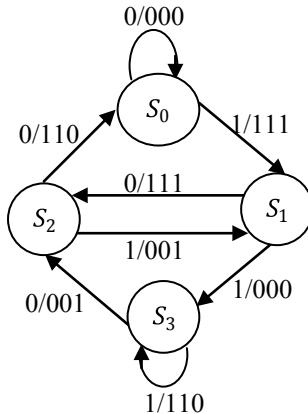
The output stream $y_3(n)$ can be represented as follows

$$y_3(n) = x(n) + x(n - 1)$$

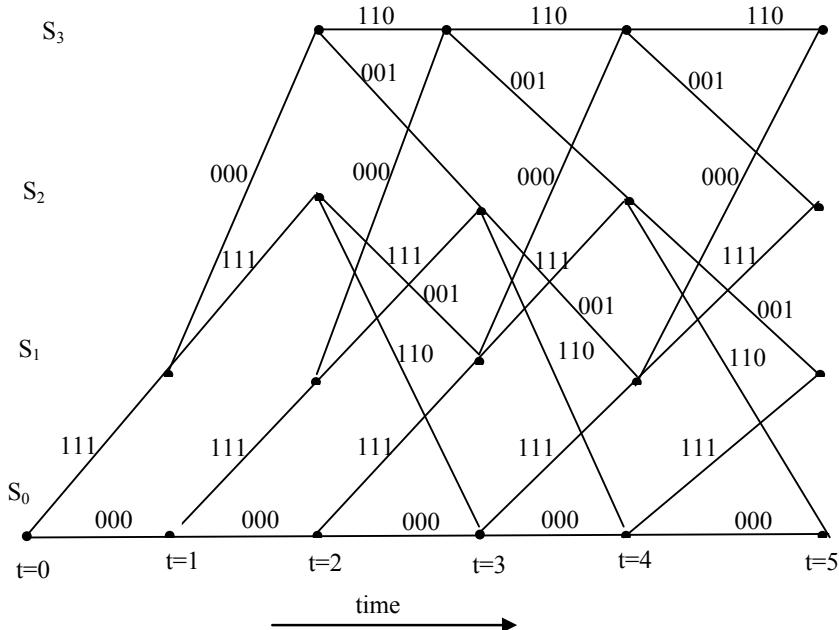
Hence, the corresponding encoder is as follows



- ii. This rate-1/3 encoder has two memory cells. So, the associated state diagram has four states as shown below.



- iii. The trellis diagram is an extension of the state diagram that explicitly shows the passage of time. The first four stages of the trellis diagram corresponding to the encoder are as follows:



5.4 Punctured Convolutional Codes

The computational complexity is an issue for implementation of Viterbi decoder for high-rate convolutional codes. This issue can be avoided by using punctured convolutional codes. The puncturing process deletes periodically selected coded bits from one or more of the output streams of a convolutional encoder. For a given fixed low-rate convolutional encoder structure, high-rate codes can be achieved by puncturing the output of low-rate convolutional encoder. The puncturing pattern is specified by a *puncturing matrix* \mathcal{P} of the form.

$$\mathcal{P} = \begin{bmatrix} \mathcal{P}_{11} & \mathcal{P}_{12} & \dots & \mathcal{P}_{1\mathcal{P}} \\ \mathcal{P}_{21} & \mathcal{P}_{22} & \dots & \mathcal{P}_{2\mathcal{P}} \\ \vdots & \vdots & \vdots & \vdots \\ \mathcal{P}_{n1} & \mathcal{P}_{n2} & \dots & \mathcal{P}_{n\mathcal{P}} \end{bmatrix} \quad (5.2)$$

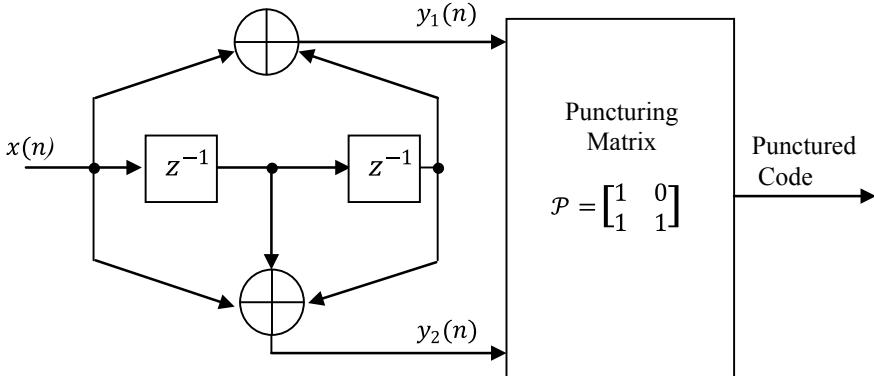


Fig. 5.10 Generation of rate 2/3 code from a rate 1/2 convolutional code

The puncturing matrix will have n rows, one for each output stream in an encoder with n output bits. The number of columns in the puncturing matrix is the number of bits over which the puncturing pattern repeats. The encoder transmits the bit corresponding to $\mathcal{P}_{ij} = 1$ and detects the bit corresponding to $\mathcal{P}_{ij} = 0$. The search for optimum punctured codes has been done by [5–7].

Example 5.9 Construct a rate 2/3 code by puncturing the output of the rate 1/2, non-systematic convolutional encoder of Example 5.6.

Solution To generate rate 2/3 code from the rate 1/2 convolutional code with constraint length 3, the puncturing matrix is

$$\mathcal{P} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

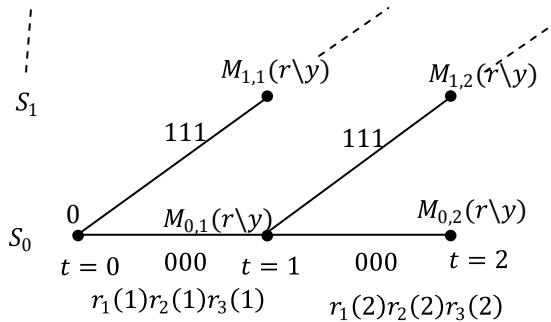
The zero entity in the second column of the second row indicates that every second bit in the output $y_1(n)$ is to be punctured. The generation of rate 2/3 code from a rate 1/2 convolutional code is shown in Fig. 5.10. The punctured encoder generates 6 code bits for every 4 message bits, and thus, the punctured code rate is 2/3.

5.5 The Viterbi Decoding Algorithm

The Viterbi algorithm is a maximum likelihood decoding algorithm for convolutional codes that involves in finding the path largest metric through the trellis by comparing the metrics of all branch paths entering each state with the corresponding received vector r iteratively. Let $S_{j,t}$ be the node corresponding to the state S_j at time t and with an assigned value $M_{j,t}(r \setminus y)$. Let m be the memory order. A distance between

the received pair of bits and branch output bits is defined as branch metric and sum of metrics of all the branches in a path is defined as path metric.

Partial path metric for a path is obtained by summing the branch metrics for the first few branches that the path traverses. For example, consider the trellis diagram of Example 5.7, the beginning of the trellis as follows



Each node in the trellis is assigned a number. This number is the partial path metric of the path that starts at state S_0 at time $t = 0$ and terminates at that node.

Let $M_{j,t}(r \setminus y)$ be the partial path metric entering the node corresponding to the state j at time t . For example, in the accompanying drawing, the label Y corresponds to the two-branch path that terminates at state S_1 at time $t = 2$. Given that the output bits corresponding to this path consist of three zeros followed by three ones, and the received sequence r with received bits of the form $r_k(t)$ indicating the k th bit in the sequence at time t .

$$M_{0,1}(r/y) = M(r_1(1)/0) + M(r_2(1)/0) + M(r_3(1)/0)$$

$$M_{1,1}(r/y) = M(r_1(1)/1) + M(r_2(1)/1) + M(r_3(1)/1)$$

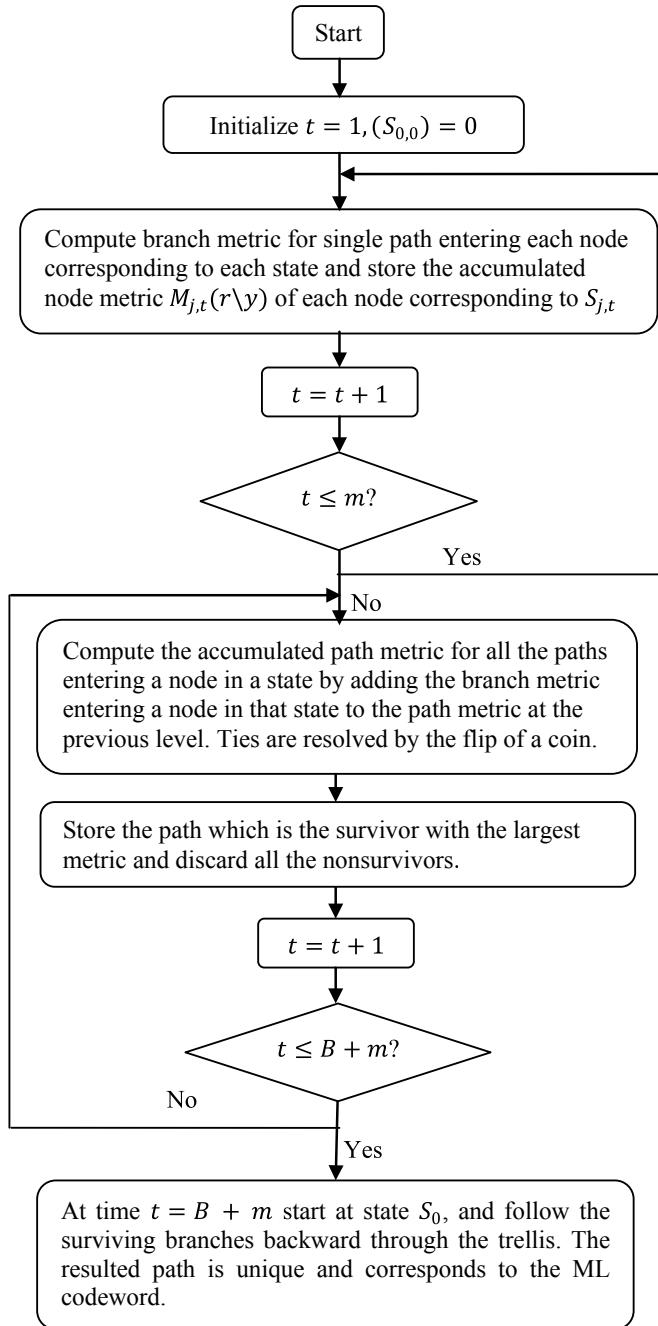
$$M_{0,2}(r/y) = M_{0,1}(r/y) + M(r_1(2)/0) + M(r_2(2)/0) + M(r_3(2)/0)$$

$$M_{1,2}(r/y) = M_{0,1}(r/y) + M(r_1(2)/1) + M(r_2(2)/1) + M(r_3(2)/1)$$

The flowchart for the iterative decoding Viterbi algorithm is shown in Fig. 5.11.

5.5.1 Hard Decision Decoding

In hard decision decoding, we will examine each received signal and make a “hard” decision to decide whether the transmitted signal is a zero or a one. These decisions form the input to the Viterbi decoder. From the decoder’s perspective and by considering the channel to be memory less, the compilation of the likelihood functions in a table is the primary step in defining the bit metrics for the channel. These conditional probabilities are first converted into log-likelihood functions and then into bit metrics.

**Fig. 5.11** Viterbi algorithm

For the BSC case shown in Fig. 1.3 of Chap. 1, the path metric is simply a Hamming distance between codeword y and received word r .

Then, the bit metric for BSC case is as follows

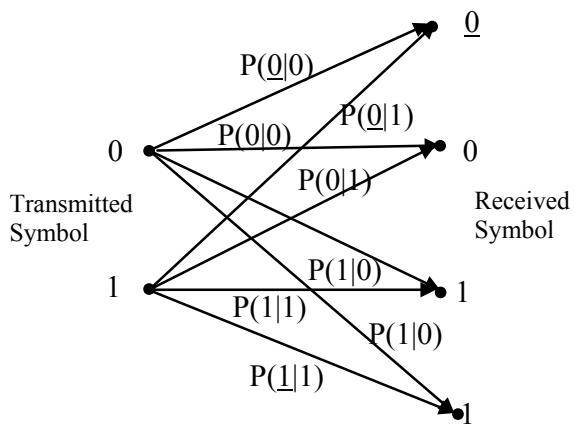
$M(r/y)$	$r = 0$	$r = 1$
$y = 0$	1	0
$y = 1$	0	1

5.5.2 Soft Decision Decoding

In soft decision decoding, “side information” is generated by the receiver bit decision circuitry and the receiver utilizes this. Instead of assigning zero or one to each received noisy binary signal as in hard decision decoding, four regions namely “strong-one,” “weak-one,” “strong-zero,” and “weak-zero” are established for soft decision decoding. Intermediate values are given to signals for which the decision is less clear. An increase in coding gain of 2–3 dB over the hard decision Viterbi decoder is provided by soft decision decoding for an additive white Gaussian noise channel.

Figure 5.12 shows a discrete symmetric channel where the underlined zero and one indicate the reception of a clear, strong signal, while the non-underlined pair denotes the reception of a weaker signal and the receiver will assign one of the four values to each received signal.

Fig. 5.12 A discrete symmetric channel model



A hard limiter makes the bit decisions in a hard decision receiver, whereas a multiple-bit analog-to-digital converter (ADC) is used in soft decision receivers for this purpose. The channel model shown in Fig. 5.12 uses a 2-bit ADC in the decision circuitry. The soft decision decoding is almost similar to the hard decision decoding but uses the increased number (and resolution) of the bit metrics.

Consider the following values for the conditional probabilities

$p(r/y)$	$r = \underline{0}$	$r = 0$	$r = 1$	$r = \underline{1}$
$y = 0$	0.50	0.25	0.15	0.05
$y = 1$	0.05	0.15	0.25	0.50

They provide the following log-likelihood values.

$\log_2 p(r/y)$	$r = \underline{0}$	$r = 0$	$r = 1$	$r = \underline{1}$
$y = 0$	-1	-2	-2.73	-4.32
$y = 1$	-4.32	-2.73	-2	-1

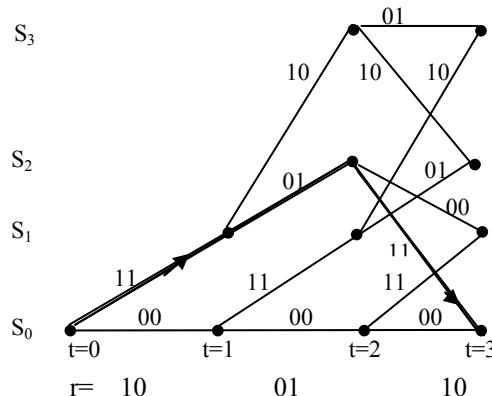
Using the expression below, we obtain a set of bit metrics that can be easily implemented in digital hardware.

$$M(r/y) = 1.5[\log_2 p(r/y) - \log_2(0.05)]$$

$M(r/y)$	$r = \underline{0}$	$r = 0$	$r = 1$	$r = \underline{1}$
$y = 0$	5	4	2	0
$y = 1$	0	2	4	5

Example 5.10 If the convolutional encoder of Example 5.6 is used to transmit code word over a BSC, and that received sequence is $r = (10, 01, 10)$, find the maximum likelihood code using Viterbi hard decision decoding algorithm

Solution



In the above figure, the maximum likelihood code word is the word corresponding to the ML path denoted by thick line in the above trellis diagram. Thus, the maximum likelihood code word is

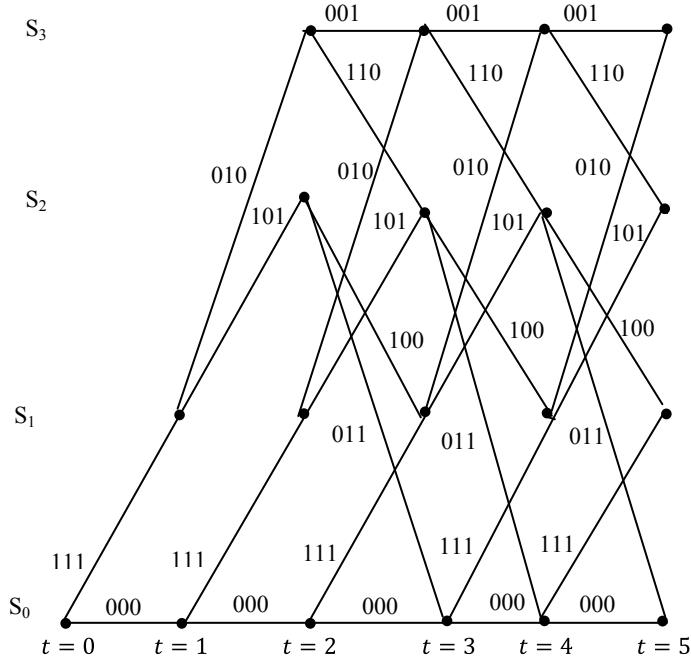
$$Y = (11, 01, 11)$$

Example 5.11 Consider the encoder shown in Fig. 5.1.

- Construct the Trellis diagram for the length-3 input sequence.
- If a code word from the encoder is transmitted over a BSC and that the received sequence is $r = (110, 110, 110, 111, 010)$, find the maximum likelihood code using Viterbi hard decision decoding algorithm.

Solution

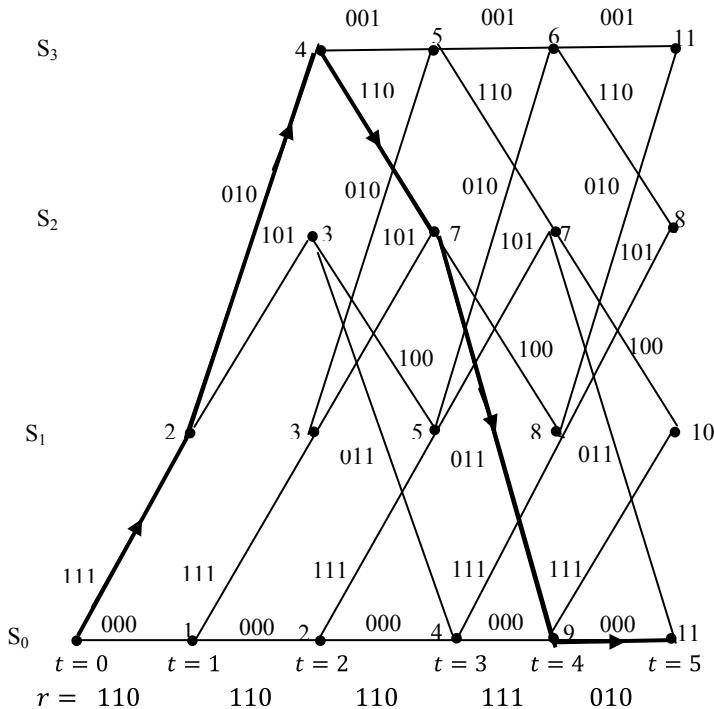
- From the state diagram shown in Fig. 5.3, for the encoder of Fig. 5.1, the following trellis diagram is constructed.



- ii. For BSC, the bit metrics chosen for hard decision are as follows

$M(r/y)$	$r = 0$	$r = 1$
$y = 0$	1	0
$y = 1$	0	1

Using the above bitmetrics and following the Viterbi decoding algorithm procedure shown in Fig. 5.11, the results of the decoding operation using hard decision decoding are shown in the following figure.



In the above figure, the maximum likelihood code word is the word corresponding to the ML path denoted by thick line in the above trellis diagram. Thus, the maximum likelihood code word is

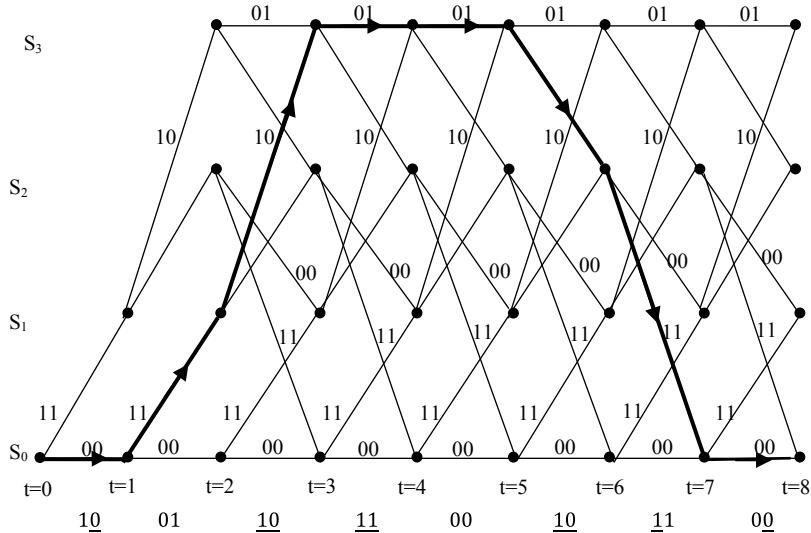
$$Y = (111, 010, 110, 011, 000)$$

Example 5.12 Considering the convolutional encoder of Example 5.6.

When the convolutional code is transmitted over a symmetric memoryless channel with the following bitmetrics shown below in the table, find the transmitted code word for the following received code word (10, 01, 10, 11, 00, 10, 11, 00) using soft decision decoding.

$M(r/y)$	$r = \underline{0}$	$r = 0$	$r = 1$	$r = \underline{1}$
$y = 0$	6	3	1	0
$y = 1$	0	1	3	6

Solution Using the above bitmetrics and following the Viterbi decoding algorithm procedure shown in Fig. 5.11, the results of the decoding operation using soft decision decoding are shown in the below figure.



In the above figure, the maximum likelihood code word is the word corresponding to the ML path denoted by thick line in the above trellis diagram. Thus, the maximum likelihood code word is $Y = (00, 11, 10, 01, 01, 10, 11, 00)$.

5.6 Performance Analysis of Convolutional Codes

5.6.1 Binary Symmetric Channel

The lower bound on the bit error rate in the convolutional codes on the binary symmetric channel with a crossover probability P is given by [8].

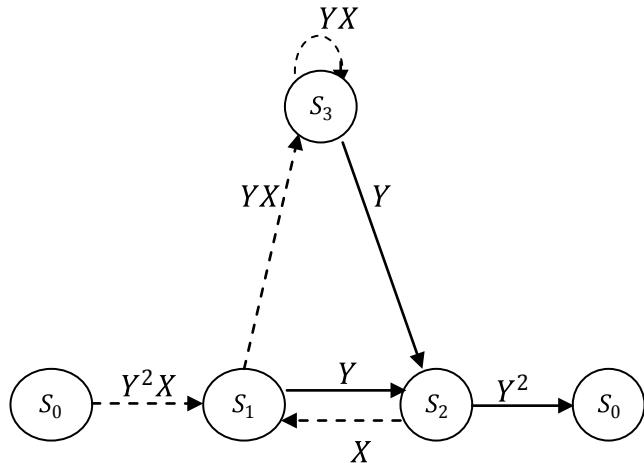
$$P_b = \begin{cases} \frac{1}{k} \sum_{k=(d_f+1)/2}^{d_f} \binom{d_f}{k} P^k (1-P)^{d_f-k} & d \text{ odd} \\ \frac{1}{2k} \binom{d_f}{d_f/2} P^{d_f/2} (1-P)^{d_f/2} + \frac{1}{k} \sum_{k=d_f/2+1}^{d_f} \binom{d_f}{k} P^k (1-P)^{d_f-k}, & d \text{ even} \end{cases} \quad (5.3)$$

Whereas the upper bound on bit error rate is given by [5].

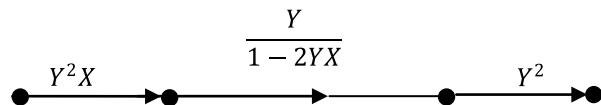
$$P_b < \frac{1}{k} \left. \frac{\partial T(Y, X)}{\partial X} \right|_{Y=2\sqrt{P(1-P)}, X=1} \quad (5.4)$$

Example 5.13 Consider the convolutional encoder from Example 5.6. Compute the upper bound and lower bound on BER for a binary symmetric channel with crossover probability $P = 0.01$.

Solution The signal-flow graph of the encoder considered in Example 5.6 can be represented as follows:



By using reduction techniques, the signal-flow graph can be simplified as



The transfer function is given by

$$T(Y, X) = \frac{Y^5 X}{1 - 2YX}$$

$$\left. \frac{\partial T(Y, X)}{\partial X} \right|_{X=1} = \frac{Y^5}{(1 - 2Y)^2}$$

Upper bound on bit error probability:

$$P < \frac{1}{k} \left. \frac{\partial T(X, Y)}{\partial y} \right|_{Y=2\sqrt{P(1-P)}, X=1} = \frac{1}{k} \left. \frac{Y^5}{(1-2Y)^2} \right|_{Y=2\sqrt{P(1-P)}},$$

Since $k = 1$ for this example

$$P_b < \left. \frac{Y^5}{(1-2Y)^2} \right|_{Y=0.198997} = 8.61 \times 10^{-4}$$

Lower bound on bit error probability: $d_f = 5$

$$\begin{aligned} p_b &= \sum_{k=3}^5 \binom{5}{k} p^k (1-p)^{5-k} = 10p^3(1-p)^2 + 5p^4(1-p) + p^5 \\ &= 9.8501 \times 10^{-6} \end{aligned}$$

5.6.2 AWGN Channel

The upper and lower bounds on the bit error rate at the output of the decoder in AWGN channel with BPSK for the unquantized soft decoding is given by

$$P_b \leq \frac{1}{k} e^{d_f E_b / N_0} Q \left(\sqrt{\frac{2d_f R_c E_b}{N_0}} \right) \left. \frac{\partial T(Y, X)}{\partial X} \right|_{Y=e^{-E_b/N_0}, X=1} \quad (5.5)$$

Since the received signal is converted to a sequence of zeros and ones before it is sent to the decoder, for hard decision decoding AWGN channel with BPSK modulation can be seen as BSC crossover probability P given by

$$P = Q \left(\sqrt{R_c \frac{2E_b}{N_0}} \right) \quad (5.6)$$

Substitution of the above P in Eq. (5.4) yields upper bound for the hard decision decoding in AWGN channel with BPSK modulation.

The coding gain of a convolutional code over an uncoded BPSK or QPSK system is upper bounded by [7].

$$\text{Coding gain} = 10 \log_{10} \left(\frac{R d_f}{2} \right) \text{dB for hard - decision} \quad (5.7a)$$

$$= 10 \log_{10}(Rd_f) \text{dB for soft - decision} \quad (5.7b)$$

Hence, the soft decision decoding introduces 3 dB increases in the coding gain over the hard decision decoding.

The BER performance of soft decision and hard decision decoding is compared through the following example.

Example 5.14 Consider the encoder used in the Example 5.12 and compare the BER performance of soft decision and hard decision decoding in an AWGN channel with BPSK modulation.

Solution The following MATLAB program 5.2 is written and used for comparison of BER performance for different E_b/N_0 using soft decision and hard decision of decoding. The comparison BER performances with an encoder used in the Example 5.12 for hard decision and soft decision decoding over an AWGN channel shown in the following Fig. 5.13.

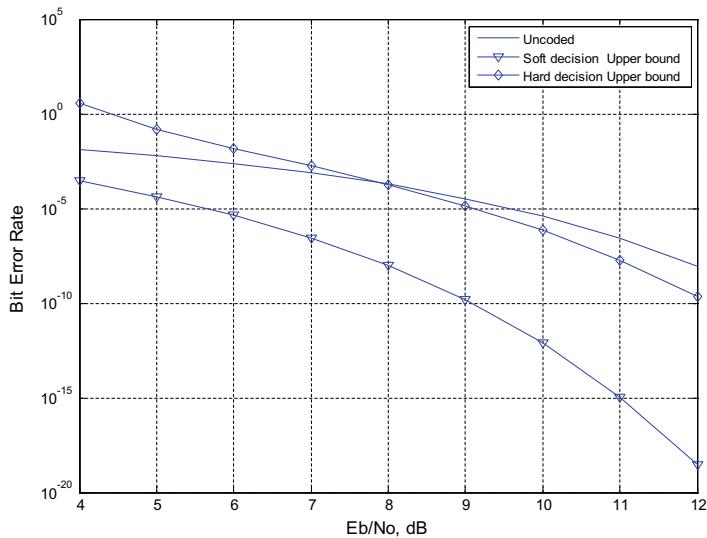


Fig. 5.13 BER performance comparisons of hard decision and soft decision decoding over an AWGN channel

Program 5.2

```

clear all; clc;
Eb_N0_dB = [4:12]; % multiple Eb/N0 values
EbN0Lin = 10.^ (Eb_N0_dB/10);
d_free=5;
for i=1:length(Eb_N0_dB)
p = 0.5* erfc ( sqrt(EbN0Lin(i)) );
Ber_uncoded(i)=p;
p = 0.5* erfc ( sqrt(0.5*EbN0Lin(i)) );
y=2*sqrt(p*(1-p));
Ber_Hd_Ub(i)=(y^5)/((1-2*y)^2);
yy=exp(-EbN0Lin(i));
yyy=(yy^5)/((1-2*yy)^2);
Ber_Sd_Lb(i)=0.5*erfc(sqrt(d_free*0.5*EbN0Lin(i)));
Ber_Sd_Ub(i)=Ber_Sd_Lb(i)*exp(d_free*EbN0Lin(i))*yyy;
end
semilogy(Eb_N0_dB,Ber_uncoded,'-'); grid on
hold on
semilogy(Eb_N0_dB,Ber_Sd_Ub,'-v');
semilogy(Eb_N0_dB,Ber_Hd_Ub,'-d');
legend('Uncoded','Soft decision Upper bound','Hard
decision Upper bound')
xlabel('Eb/No, dB');
ylabel('Bit Error Rate');

```

From Fig. 5.13, it is observed that soft decision decoding offers 3 dB increasing coding gain in over hard decision which satisfies the Eq. (5.7a) and (5.7b).

5.6.3 Rayleigh Fading Channel

The union upper bound on the bit error probability for better BER estimate for convolutional codes is given by [9].

$$P_b < \sum_{d=d_f}^{\infty} c_d P_d \quad (5.8)$$

where c_d is the information error weight for error events of distance d , and d_f is the free distance of the code. P_d is the pairwise error probability, given by [9]. For an AWGN channel, P_d is given by

$$P_d = Q\left(\sqrt{2dR\frac{E_b}{N_0}}\right) \quad (5.9)$$

where R is the code rate, E_b is received energy per information bit, N_0 is the double-sided power spectral density of the noise.

The pairwise error probability in a Rayleigh fading channel is given by [9].

$$P_d = (P_e)^d \sum_{k=0}^{d-1} \binom{d-1+k}{k} (1-P_e)^k \quad (5.10)$$

where $P_e = \frac{1}{2} \left(1 - \sqrt{\frac{\gamma_b R}{1+\gamma_b R}} \right)$.

Where γ_b is the average $\frac{E_b}{N_0}$.

A comparison of the upper bound on the BER in the AWGN and flat Rayleigh fading channels for ODS convolutional codes [9] with $R = 1/4$ and constraint length of seven is obtained using the MATLAB program 5.3 and shown in Fig. 5.14.

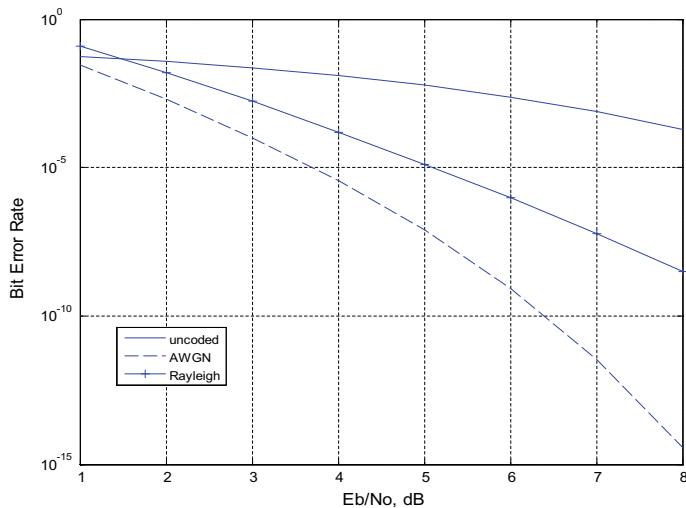


Fig. 5.14 A comparison of the upper bound on the BER in the AWGN and flat Rayleigh fading channels ODS convolutional codes with $R = 1/4$ and constraint length of 7

Program 5.3

```

clear all;clc;
R=0.25;
pd2=[];
for i=1:8
    Eb_N0_dB=i;
    pb=0;pbr=0;
    d=20;
    for j=1:20
        cd=[];
        cd=[3;0;17;0;32;0;66;0;130;0;364;0;889;0;1975;0;5168;0;111
        13;0];
        EbNOLin=10.^((Eb_N0_dB)/10);
        q1=sqrt((EbNOLin*R)/(1+(EbNOLin*R)));
        q=0.5*(1-q1);pd1=0;
        for k=0:(d-1)
            pd1=pd1+nchoosek((d-1+k),k)*((1-q)^k);
        end
        pd=(q^d)*pd1;
        pd1=0.5*erfc(sqrt(d*R*EbNOLin));
        pd2(j)=pd1; pdr(j)=pd;
        pb=pb+cd(j)*pd2(j);
        pbr=pbr+cd(j)*pdr(j);
        d=d+1;
    end
    berawg(i)=pb;
    berrayf(i)=pbr;
    p = 0.5* erfc ( sqrt(EbNOLin) );
    Ber_uncoded(i)=p;
end
semilogy(1:8,Ber_uncoded,'-'); grid on
hold on
semilogy(1:8,berawg,'--');
semilogy(1:8,berrayf,'-+');
legend('uncoded','AWGN','Rayleigh ')
 xlabel('Eb/No, dB'); ylabel('Bit Error Rate');

```

5.7 Tail-Biting Convolutional Code

A convolutional code is generated by a finite-state machine which has states that depend on the current and past inputs. Normally, the encoder starts and ends in the all-zeros state. A tail of zeros can be used to bring the encoder back to the all-zeros state, and the decoder is more effective when it knows the starting and ending states. In zero tailing method, the encoder's states are assumed to start in all-zeros state. Once all the input information sequence is passed through the encoder, it is not in all-zeros state. In order to get back the encoder into all-zeros state, zeros equal to m , i.e., number of memory elements are fed into the encoder. This results in a fractional rate loss.

Consider an $(n; k; m)$ encoder, then m zeros will be padded for each of the k input sequences of length L producing $n(L + m)$ output bits. Then, the effective rate, i.e., average number of input bits carried by an output bit is given as [10].

$$R_{\text{eff}} = \frac{kL}{n(L + m)} = R \frac{L}{L + m}$$

$\frac{L}{L+m}$ is the fractional rate loss. Hence, the disadvantage is that extra bits need to be transmitted at the cost of extra transmission time, and as a consequence, a larger E_b/N_0 is required for a given probability of error.

5.7.1 Tail-Biting Encoding

Tail-biting convolutional codes overcome the rate loss caused by the encoding of convolutional codes by zero tailing. With a tail-biting convolutional code [11], the memory elements are initialized by the last m input bits of the input sequence in order to be sure that encoder ends in the same state.

Tail-biting encoding tries to overcome the need to transmit extra termination bits. In tail-biting encoding, the encoder's m shift registers are initialized with the last m bits of the input information sequence. Output obtained during the initialization is not transmitted across the channel. This method ensures that the encoder start and end state are same.

Example 5.15 Consider the convolutional encoder of Example 5.6. Obtain encoded sequence from tail-biting encoding process for input $x(n) = [1 \ 1 \ 0 \ 1]$.

Solution The transitions for the encoder of Example 5.6 are shown in Table 5.2.

Table 5.2 Transitions for convolutional encoder of Example 5.6

$x(n)$	State at n	State at $n + 1$	Output
0	S_0 (00)	S_0	00
1		S_1	11
0	S_1 (10)	S_2	01
1		S_3	10
0	S_2 (01)	S_0	11
1		S_2	00
0	S_3 (11)	S_2	10
1		S_3	01

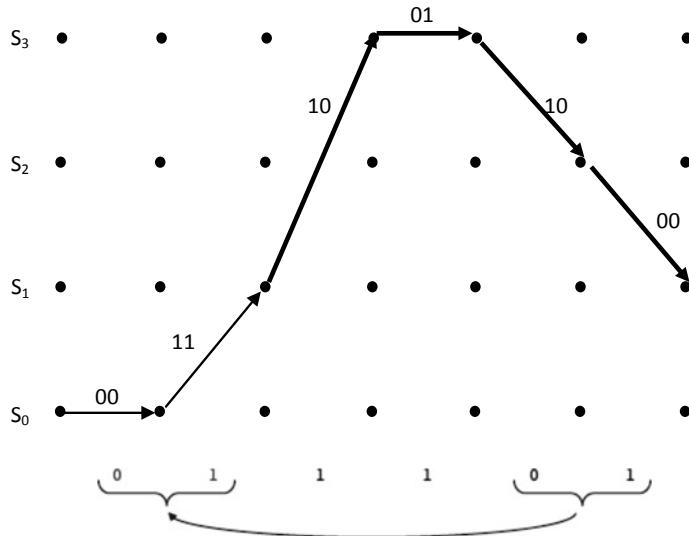


Fig. 5.15 Trellis diagram of example for input $x(n) = [1 \ 1 \ 0 \ 1]$. With tail biting

Step 1: Since the last two bits are 01, the final state is 10. Thus, the final state is S_1 . Then, the initial state is set as S_1 , as shown in the trellis diagram of the convolutional encoder shown in Fig. 5.15.

Step 2: copy the last 2 data bits to the beginning of the message. These bits were used only to determine the initial state, do not transmit the associated code bits as shown in the trellis diagram for encoding the input sequence as shown in Fig. 5.15.

Thus, the code word due to tail-biting process corresponding to the input $x(n) = [1 \ 1 \ 0 \ 1]$ is $Y = (10, 01, 10, 00)$.

5.7.2 Tail-Biting Encoding Using MATLAB

The tail-biting convolution encoding is illustrated through the following example.

Example 5.16 Consider the following 802.16e standard tail-biting convolutional encoder. Obtain the encoded sequence due to tail-biting encoding process for randomly generated input sequence $x(n)$ (Fig. 5.16).

Solution The generators for the considered encoder are

$$g_1 = [1 \ 1 \ 1 \ 1 \ 0 \ 0 \ 1]; g_2 = [1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1]$$

The following MATLAB program 5.4 is written and used to obtain the encoding sequence due to tail-biting process.

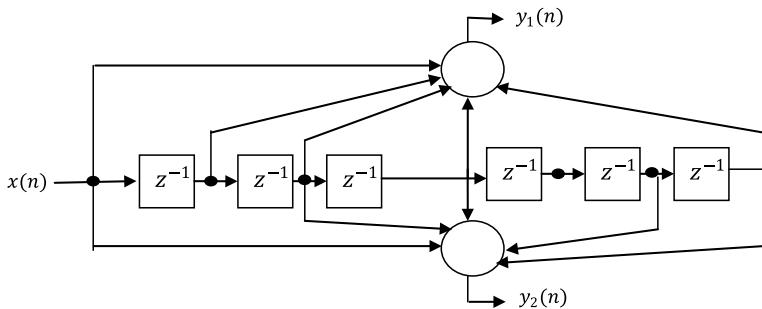


Fig. 5.16 802.16e standard tail-biting convolutional encoder

Program 5.4

```

clear all;clc;
input_msg = randi([0 1],10,1);
g1= str2num(dec2base(bin2dec('1111001'),8))
g2= str2num(dec2base(bin2dec('1011011'),8))
% trellStr = poly2trellis(7, [133 171 165]);%LTE
trellStr = poly2trellis(7, [ g1 g2 ]);%802.16e Standard
k = log2(trellStr.numInputSymbols);
m =log2(trellStr.numStates);
%% Collect tail bits
tail_bits = input_msg(end + 1 - k*m:end);
%% Obtain state bits
hEnc1 = comm.ConvolutionalEncoder;
hEnc1.TrellisStructure = trellStr;
hEnc1.TerminationMethod = 'Truncated';
hEnc1.FinalStateOutputPort = true;
[states, fin_state] = step(hEnc1, tail_bits);
%% Define main encoder
hEnc = comm.ConvolutionalEncoder;
hEnc.TrellisStructure = trellStr;
hEnc.TerminationMethod = 'Truncated';
hEnc.InitialStateInputPort = true;
coded_data = step(hEnc, input_msg, fin_state);

```

Random input sequence: [1 1 0 1 1 0 0 1 1].

Encoded sequence: [1 0 0 1 1 1 0 1 0 0 1 1 0 1 1 0 0 0 0].

5.7.3 Tail-Biting Decoding

In tail-biting codes, the encoder starts and ends in the same state, but the state remains unknown to the decoder. The advantage of tail-biting convolutional codes is that overhead of the framed code with tail can be avoided. A Viterbi decoder with a known

starting and ending state can be achieved by making number of known information bits terminate the convolutionally encoded sequences. Disadvantage of this method is the requirement of overhead due to which there is a fractional rate loss. Consider an $(n; k; m)$ convolutional code with rate $R = k/n$, number of memory elements m . Then, when N is the length of the encoded codeword, the rate is

$$R = \frac{k}{n} \frac{N - m}{N}$$

When $N \gg m$, i.e., for long blocks, the rate loss is small.

A tail-biting trellis can be visualized as a cylinder by connecting the starting and ending states. The Viterbi algorithm can be run on the cylinder [11]. In theory, the algorithm would have to run forever by cycling around the cylinder. In practice, it is sufficient to limit the cycling around the cylinder. The wrap depth is the amount of trellis on the cylinder that is traversed more than once. It achieves maximum likelihood decoding performance as the number of cycle's approaches infinity. Performance of decoding depends on the length of the trellis. The decoder is not limited to number of cycles while decoding and must satisfy bound specifying the number of errors needed to be corrected.

Consider an $(n; k; m)$ tail-biting convolutional code with rate $1/n$. Let the length of the input information bits be L and length of codeword be $N = nL$. Circular decoding uses the Viterbi decoding trellis of length $H + L + T$ where H and T are the head decoding length and tail decoding length, respectively. The starting nT bits of a received sequence are added at the end and the last nH bits are appended at the beginning of the received sequence.

Example 5.17 Draw the Viterbi decoding trellis diagram the tail-biting encoding process of Example 5.14 with wrap depth is equal to 3.

Solution The stepwise procedure to draw the Viterbi decoding trellis diagram is as follows:

- Step 1: Expand trellis by 3 sections before and after code sequence pend first received pairs.
- Step 2: Append _rst 3 received pairs to end of sequence.
- Step 3: Prepend last 3 received pairs to start of sequence.

Applying the above three steps, the Viterbi decoding trellis obtained for the tail-biting encoding process of Example 5.13 is shown in Fig. 5.17.

5.8 Performance Analysis of Tail-Biting Convolutional Codes

The BER performance of tail-biting convolutional codes over AWGN channel is illustrated through the following two examples using MATLAB.

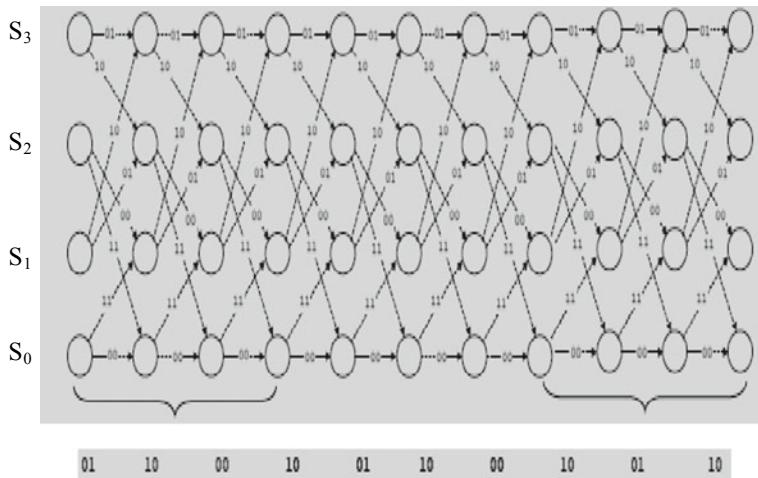


Fig. 5.17 Viterbi decoding trellis of Example 5.14

Example 5.18 Consider IEEE 802.16e standard tail-biting convolutional encoder of rate 1/2 as shown in Fig. 5.16. Evaluate its BER performance using BPSK modulation over AWGN channel for information block length 100 bits.

Solution Using MATLAB commands commterminatedcnv, commtailbiting, and bertool, BER performance using BPSK modulation over AWGN channel for information block length 100 bits is evaluated and shown in Fig. 5.18.

Example 5.19 Consider an LTE convolutional encoder of rate 1/3 with constraint length $K = 7$ as shown in Fig. 5.19. Evaluate its BER performance using QPSK modulation.

Solution For the given encoder, the generator matrices $g_1 = 171_{\text{oct}} = [1 \ 1 \ 1 \ 1 \ 0 \ 0 \ 1]$, $g_2 = 133_{\text{oct}} = [1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1]$, $g_3 = 165_{\text{oct}} = [1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1]$. The following MATLAB program 5.5 is written and used to obtain the bit error rate (BER), of QPSK over AWGN channel with no tail biting. The E_b/N_0 versus BER plot is shown in Fig. 5.20.

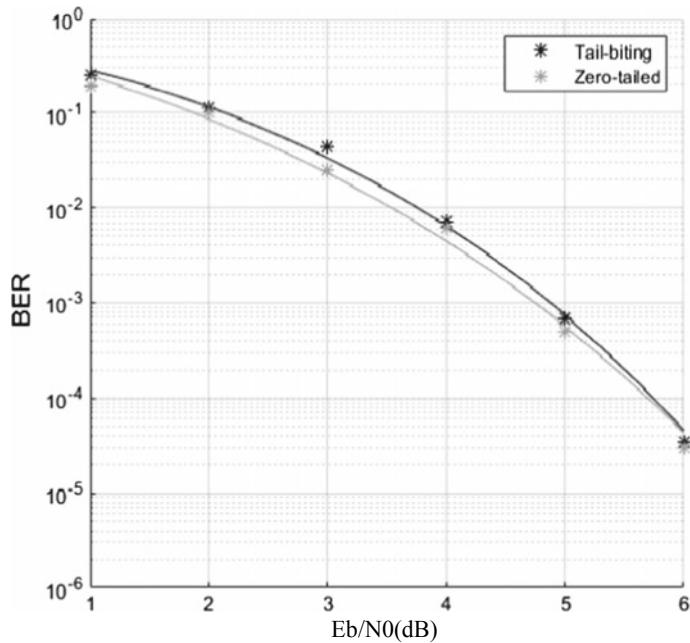


Fig. 5.18 BER performance of 802.16e standard tail-biting convolutional encoder

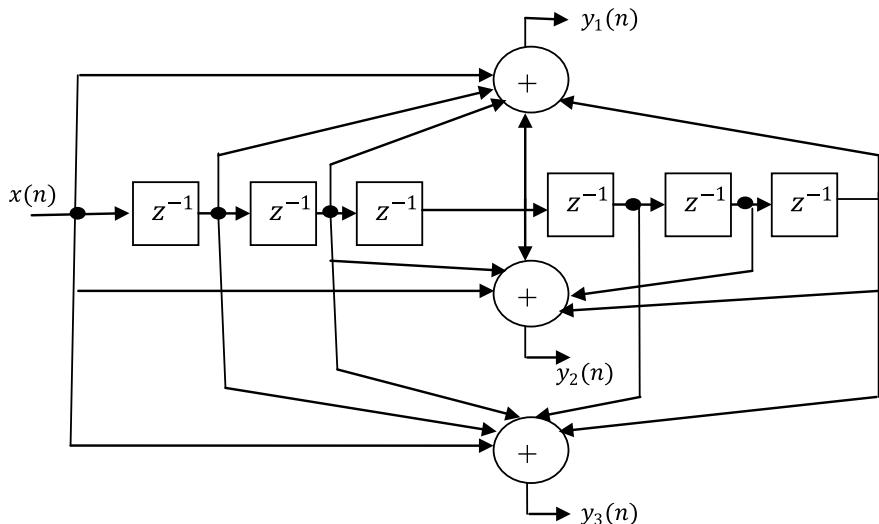


Fig. 5.19 LTE convolutional encoder

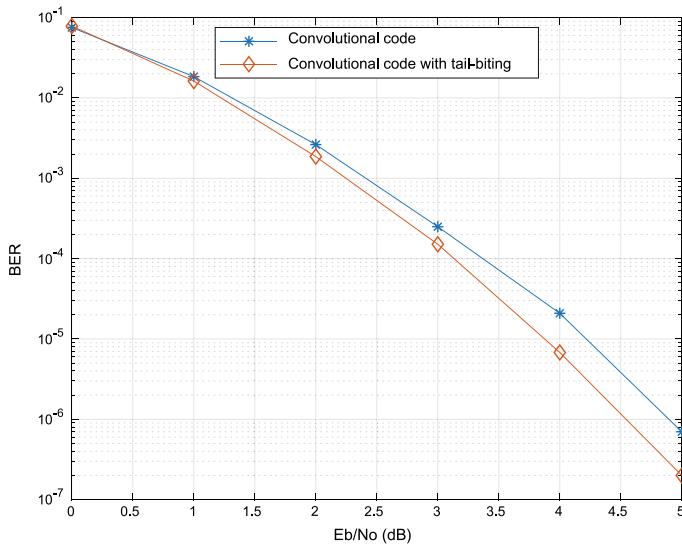


Fig. 5.20 Viterbi decoding trellis of Example 5.14

Program 5.5

```

clear; close all
rng default
M = 4;% 64;           % Modulation order
k = log2(M);          % Bits per symbol
EbNo = (0:10)';       % Eb/No values (dB)
numSymPerFrame = 1000; % Number of QAM symbols per frame
berEstHard = zeros(size(EbNo));
g1= str2num(dec2base(bin2dec('1011001'),8));
g2= str2num(dec2base(bin2dec('1111001'),8));
g3= str2num(dec2base(bin2dec('1110101'),8));
trellis = poly2trellis(7, [ g1 g2 g3 ]);%LTEtrellis = poly2trellis(7,[133 171 165]);
tbl = 32;
rate = 1/3;
for n = 1:length(EbNo)
    snrDB = EbNo(n) + 10*log10(k*rate); % Convert Eb/No to SNR
    noiseVar = 10.^(-snrDB/10); % Noise variance calculation
    % Reset the error and bit counters
    [numErrsIte,numErrsSoft,numBits] = deal(0);
    while numBits < 1e7
        dataIn = randi([0 1],numSymPerFrame*k,1); % Generate binary data and convert to symbols
        dataEnc = convenc(dataIn,trellis); % encode the data
        codedData = lteConvolutionalEncode(dataIn); % encode the data with tail-biting
        txSymbols = lteSymbolModulate(codedData,'QPSK');
        txSig = qammod(dataEnc,M,'InputType','bit','UnitAveragePower',true); % QAM modulate
    end
    berEstHard(n) = 1 - sum(txSig==0)/tbl;
end

```

```

rxSig = awgn(txSig,snrdB,'measured'); % Pass through AWGN channel
rxSymbols=awgn( txSymbols,snrDB,'measured');
rxDataSoft = qamdemod(rxSig,M,'OutputType','approxllr', ...
'UnitAveragePower',true,'NoiseVariance',noiseVar); % Demodulate the noisy signal
softBits = lteSymbolDemodulate(rxSymbols,'QPSK','Soft');
rxBits = lteConvolutionalDecode(softBits);
dataSoft = vitdec(rxDataSoft,trellis,tbl,'cont','unquant'); % Viterbi decode dat
numErrsInFrameSoft = biterr(dataIn(1:end-tbl),dataSoft(tbl+1:end)); % Calculate the number
of bit errors in the frame
numErrsInFramelte=biterr(dataIn,rxBits);
% Increment the error and bit counters
numErrsSoft = numErrsSoft + numErrsInFrameSoft;
numErrsLTE=numErrsLTE+numErrsInFrameLTE;
numBits = numBits + numSymPerFrame*k;
end
% Estimate the BER for both methods
berEstSoft(n) = numErrsSoft/numBits;
berEstLTE(n) = numErrsLTE/numBits;
semilogy(EbNo, berEstSoft, '-*')
hold on
semilogy(EbNo,berEstLTE, '-d')
legend('Convolutional code with tail-biting','location','best')
grid
xlabel('Eb/No (dB)')
ylabel('BER')

```

5.9 Problems

1. Consider the encoder shown in Fig. 5.21 and determine the output code word using D -transform for the input sequence $x(n) = (1001)$.
2. Consider the encoder shown in Fig. 5.21 and
 - i. Draw the state diagram for the encoder.
 - ii. Draw the trellis diagram for the encoder.
 - iii. Find the transfer function and the free distance of the encoder.
3. Consider the encoder shown in Fig. 5.22

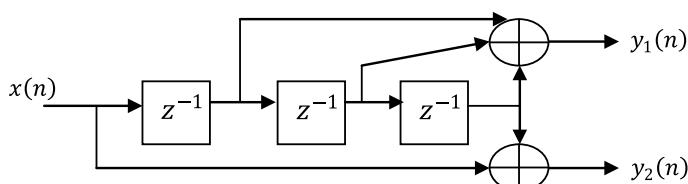
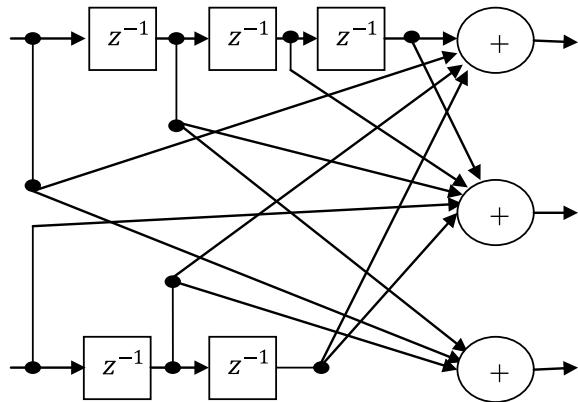


Fig. 5.21 A rate-1/2 convolutional encoder

Fig. 5.22 A rate-1/2 convolutional encoder



- i. Find impulse response.
- ii. Find the transfer function matrix.
- iii. Use the transfer function matrix to determine the code word associated with the input sequence $x = (11, 10, 01)$.
4. Consider an encoder with impulse responses $g_1 = (1111)$ and $g_2 = (1111)$. Determine whether the encoder generates a catastrophic convolutional code.
5. Construct a rate 3/4 code by puncturing the output of the rate 1/3, for systematic convolutional encoder shown in Fig. 5.2. And draw the trellis diagram of the punctured code.
6. Consider the convolutional encoder of Example 5.6. Obtain encoded sequence from tail-biting encoding process for input $x(n) = [1 \ 1 \ 0 \ 1 \ 1 \ 0]$.
7. If the convolutional encoder of Example 5.6 is used to transmit code word over a BSC, and that received sequence is $r = (10, 01, 10, 11)$, find the maximum likelihood code using Viterbi hard decision decoding algorithm.
8. If a code word from the encoder of Example 5.8 is transmitted over a BSC and that the received sequence is $r = (101, 100, 001, 011, 111, 101, 111, 110)$, find the maximum likelihood code using Viterbi hard decision decoding algorithm.
9. If a code word from the encoder of Example 5.8 is transmitted over a BSC and that the received sequence is $r = (\underline{101}, \underline{100}, \underline{001}, \underline{011}, \underline{110}, 110, \underline{111}, \underline{110})$, find the maximum likelihood code using Viterbi soft decision decoding algorithm.

$M(r y)$	$r = \underline{0}$	$r = 0$	$r = 1$	$r = \underline{1}$
$y = 0$	5	4	2	0
$y = 1$	0	2	4	5

10. Draw the Viterbi decoding trellis diagram the tail-biting encoding process of Problem 6 with wrap depth is equal to 3.

5.10 MATLAB Exercises

1. Write a MATLAB program to simulate BER performance of a convolutional encoder of your choice using hard decision and soft decision decoding over an AWGN channel and comment on the results.
2. Write a MATLAB program to simulate BER performance of a convolutional encoder of your choice using soft decision decoding over an AWGN and Rayleigh fading channel and comment on the results.

References

1. J.L. Massey, M.K. Sain, Inverse of linear sequential circuits. *IEEE Trans. Comput.* **C-17**, 330–337 (1968)
2. Heller, J.A., Short constraint Length Convolutional Codes, vol. 3 (Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA Space Program Summary 37–54, 1968), pp. 171–174
3. D.J. Costello, TR., Free Distance Bounds for Convolutional Codes, in *IEEE Transactions on Information Theory*, vol. IT-20, No. 3 (1974), pp. 356–365
4. G.D. Forney Jr., Convolutional codes II: maximum likelihood decoding. *Inf. Control* **25**, 222–266 (1974)
5. J. Cain, G. Clark, J. Geist, Punctured convolutional codes of rate $(n - 1)/n$ and simplified maximum likelihood decoding. *IEEE Trans. Inf. Theory* **IT-25**(1), 97–100 (1979)
6. Y. Yasuda, K. Kashiki, Y. Hirata, High-rate punctured convolutional codes for soft decision Viterbi decoding. *IEEE Trans. Commun.* **3**, 315–319 (1984)
7. K. Hole, New short constraint length rate $(N - 1)/N$ punctured convolutional codes for soft decision Viterbi decoding. *IEEE Trans. Commun.* **9**, 1079–1081 (1988)
8. Stephen B. Wicker, *Error Control Systems for Digital Communication and Storage* (Prentice Hall, New Jersey, 1995)
9. Pal Franger, Pal Orten, Tony Ottosson, Convolutional codes with optimum distance spectrum. *IEEE Commun. Letters* **3**(11), 317–319 (1999)
10. J. Lin, D.J. Costello Jr., *Error control coding: fundamentals and applications*, 2nd edn. (Upper Saddle River, Pearson Prentice Hall, NJ, 2004)
11. C. Matthew, A. Valenti, Tutorial on Channel Coding for IEEE 802.16e Mobile WiMAX, *IEEE ICC* (2009)

Chapter 6

Turbo Codes



The groundbreaking codes called turbo codes are introduced in [1, 2]. The best-known convolutional codes are mostly non-systematic. However, in turbo encoders, systematic convolutional codes are used. Turbo codes are generated by using the parallel concatenation of two recursive systematic convolutional (RSC) encoders. This chapter discusses turbo encoding, iterative turbo decoding, and performance analysis of turbo codes.

6.1 Non-recursive and Recursive Systematic Convolutional Encoders

A convolutional code is said to be systematic if the input sequence is reproduced unaltered in the output code word. The following rate-1/2 convolutional encoder is an example for a systematic convolutional encoder (Fig. 6.1).

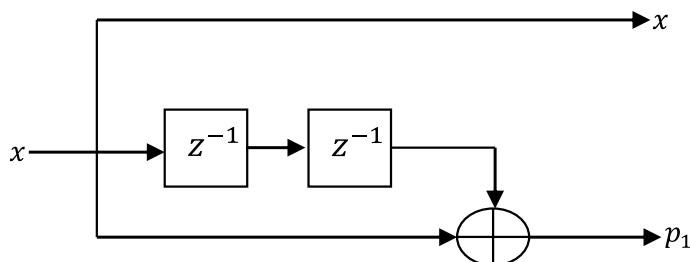


Fig. 6.1 Non-recursive systematic convolutional (SC) encoder

6.1.1 Recursive Systematic Convolutional (RSC) Encoder

Consider the conventional convolutional encoder with rate $\frac{1}{2}$ and constraint length 3 as shown in Fig. 6.2.

The generator sequences of the above non-recursive non-systematic encoder are $g_1 = [1 \ 1 \ 1]$ and $g_2 = [1 \ 0 \ 1]$.

The state diagram representation of the above non-recursive encoder is shown in Fig. 6.3.

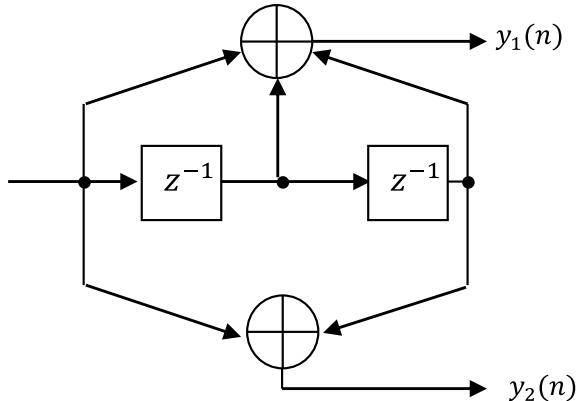


Fig. 6.2 Non-recursive non-systematic convolutional (NSC) encoder

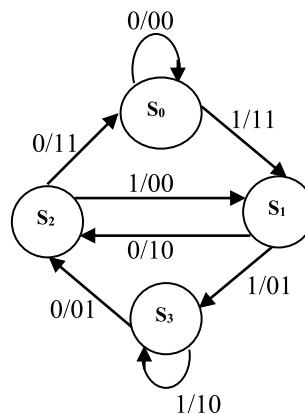


Fig. 6.3 NSC encoder state diagram

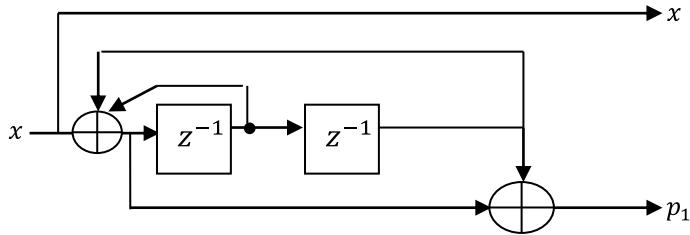


Fig. 6.4 Recursive systematic convolutional (RSC) encoder

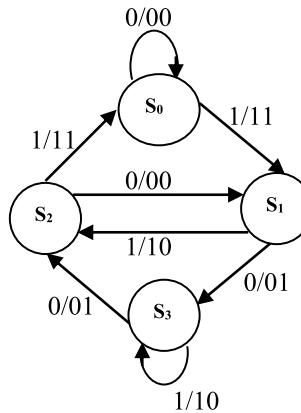


Fig. 6.5 RSC encoder state diagram

The equivalent RSC encoder of the non-recursive non-systematic encoder of Fig. 6.2 is shown in Fig. 6.4. It is obtained by feeding back the contents of the memory elements to the input with the generator function $G = \begin{bmatrix} 1 & g_2 \\ g_1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1+D^2 \\ 1+D+D^2 & 1 \end{bmatrix}$.

The state diagram of the above RSC encoder is shown in Fig. 6.5.

From Figs. 6.3 and 6.5, it is clear that the state diagrams of the NSC and RSC encoders are very similar. Further, both the codes have the same minimum free distance and trellis structure. Hence, the first event error probability is same for both the codes; however, bit error rates (BERs) are different as BER depends on the encoder's input-output correspondence. At low signal-to-noise ratios E_b/N_0 , the BER for a RSC code is lower than that of the corresponding NSC code.

6.2 Turbo Encoder

A turbo encoder structure consists of two identical RSC encoders in parallel concatenation as shown in Fig. 6.6. It is a rate 1/3 encoder.

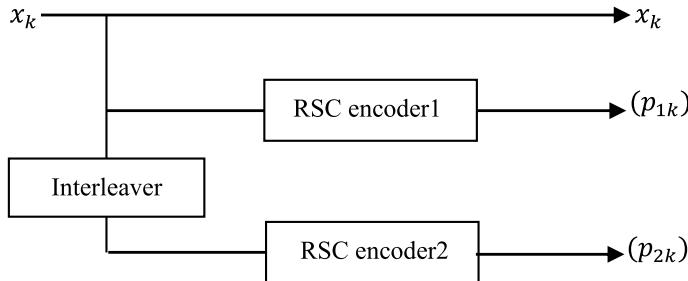


Fig. 6.6 Turbo encoder

The two RSC encoders work synergistically in parallel. The RSC encoder 1 takes the data bits x and produces a low-weight parity bits (p_{1k}) from them. The RSC encoder 2 gets the data bits x scrambled by an interleaver and computes high-weight parity bits (p_{2k}) from the scrambled input bits. Thus, moderate weight turbo code is generated with combined low-weight code from encoder 1 and high-weight code from encoder 2. Finally, the original input sequence x along with the two strings of parity bits is transmitted over the channel.

6.2.1 *Different Types of Interleavers*

The BER performance of turbo codes can be improved significantly by using interleaver as it affects the distance properties of the code by avoiding low-weight codewords [3].

Block Interleaver

The block interleaver is one of the most frequently used types of interleavers in communication systems. It fills a matrix with the input data bit stream row by row and then sends out the contents column by column. A block interleaver is shown in Fig. 6.7. It writes in $[0\ 0 \dots 1\ 0\ 1 \dots 0 \dots 1 \dots 1\ 0\ 1 \dots 0\ 1]$ and reads out $[0\ 1 \dots 1\ 0\ 0 \dots 0\ 0\ 0 \dots 1\ 1]$.

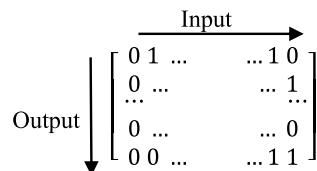


Fig. 6.7 Block interleaver

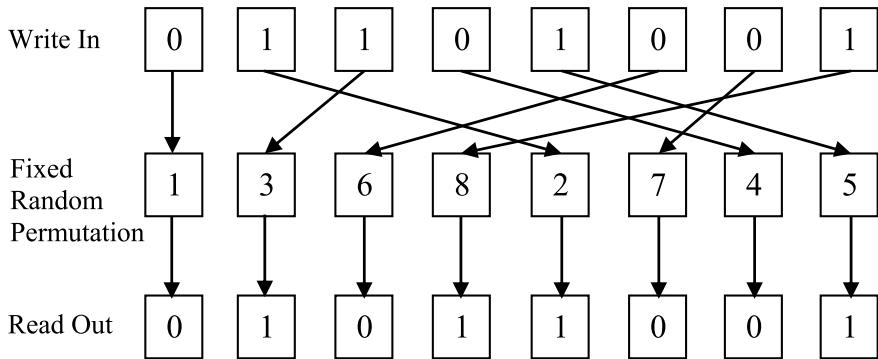


Fig. 6.8 Random interleaver

Pseudorandom Interleaver

A random interleaver maps the input sequence according to the permutation order using a fixed random permutation. A random interleaver with input sequence of length 8 is shown in Fig. 6.8.

6.2.2 Turbo Coding Illustration

We consider the following numerical examples to illustrate turbo coding. The operation of the encoders used in these examples is characterized by the corresponding trellis diagram.

Example 6.1 For the turbo encoder shown below, find the output codeword for the input data sequence $x = \{1\ 1\ 0\ 0\}$ assuming the RSC encoder 1 trellis is terminated. Let the interleaver be $\{5, 2, 4, 0, 1, 3\}$ (Fig. 6.9).

Solution The two binary memory elements can assume any one of four states

$S_0 = 00; S_1 = 10; S_2 = 01; S_3 = 11, state = [y(n-1)y(n-2)]$ as shown in Table 6.1. The trellis diagram corresponding to Table 6.1 is shown in Fig. 6.10.

The input sequence is fed to the RSC encoder 1. The resultant path through the trellis is shown in Fig. 6.11.

Now, the input is fed through the following pseudorandom interleaver shown in Fig. 6.12.

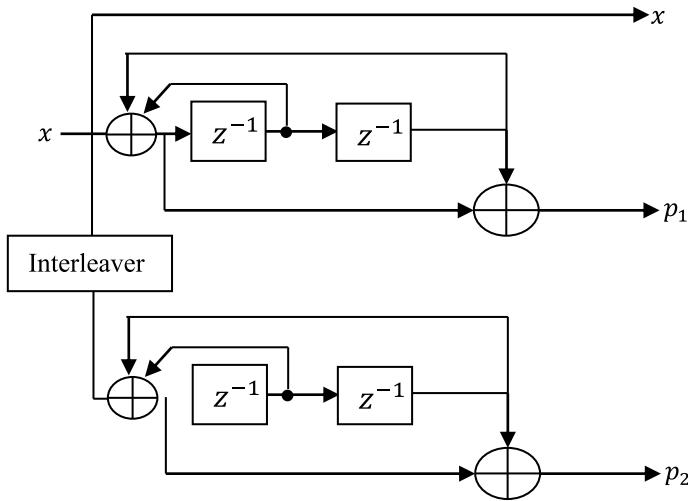


Fig. 6.9 Turbo encoder of Example 6.1

Table 6.1 Transitions for turbo encoder of Example 6.1

$x(n)$	State at n	State at $n + 1$	$P_1(n)$
0	S_0 (00)	S_0	0
1		S_1	1
0	S_1 (10)	S_3	1
1		S_2	0
0	S_2 (01)	S_1	0
1		S_0	1
0	S_3 (11)	S_2	1
1		S_3	0

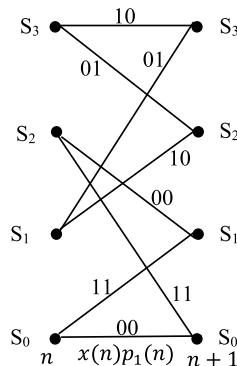


Fig. 6.10 Trellis diagram for encoder of Example 6.1

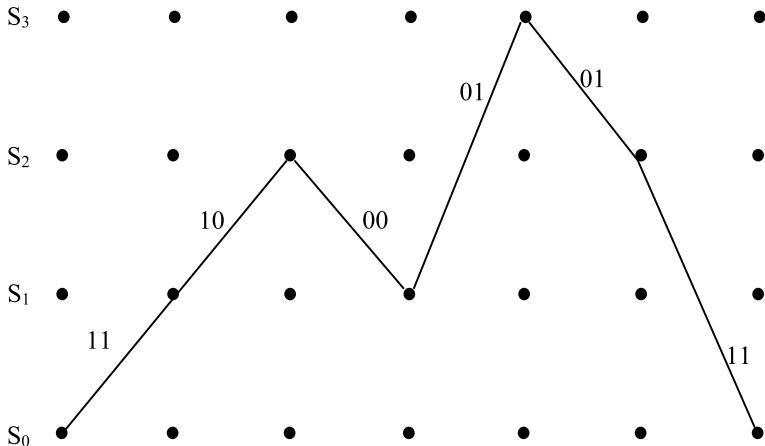


Fig. 6.11 Trellis path corresponding to input sequence of Example 6.1

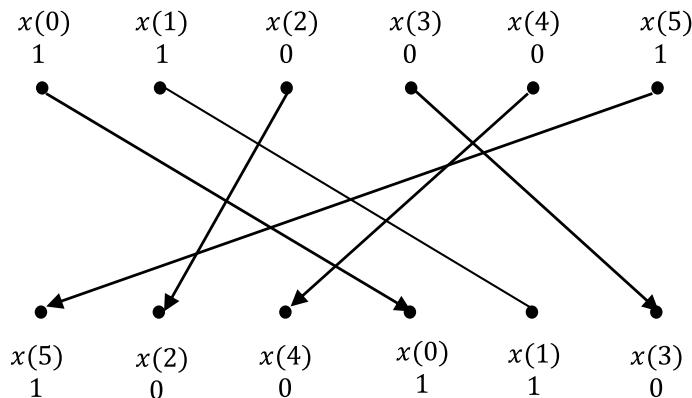


Fig. 6.12 Pseudorandom interleaver of Example 6.1

The block of permuted data bits is then fed into RSC encoder 2, resulting in the path through the trellis is shown in Fig. 6.13.

The encoder output data bits and the parity bits are mapped to symbols as shown in Table 6.2.

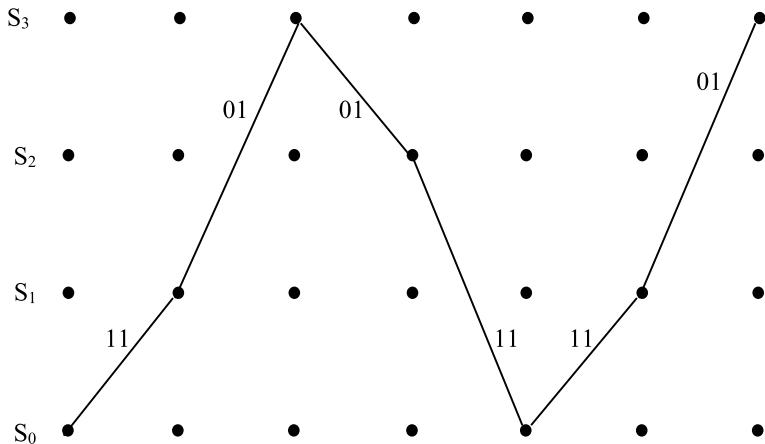


Fig. 6.13 Trellis path corresponding to interleaved input sequence of Example 6.1

Table 6.2 Output of encoder of Example 6.1

$x(n)$	$p_1(n)$	$p_2(n)$	\rightarrow	$x(n)$	$p_1(n)$	$p_2(n)$
1	1	1		1	1	1
1	0	1		1	-1	1
0	0	1		-1	-1	1
0	1	1		-1	1	1
0	1	1		-1	1	1
1	1	1		1	1	1

Example 6.2 The UMTS (Universal Mobile Telecommunications System) standard turbo-encoder with RSC encoder generator function $G = \left[1 - \frac{1+D+D^3}{1+D^2+D^3} \right]$ is shown in Fig. 6.14, find the output codeword for the input data sequence $x = \{1\ 1\ 0\ 0\}$ assuming RSC encoder 1 trellis is terminated. Let the interleaver be $\{2, 6, 4, 5, 0, 1, 3\}$.

Solution The three binary memory elements can assume any one of eight states $S_0 - 000; S_1 - 100; S_2 - 010; S_3 - 110; S_4 - 001; S_5 - 101; S_6 - 011'; S_7 - 111$. as shown in Table 6.3. The trellis diagram corresponding to Table 6.3 is shown in Fig. 6.15.

The input sequence is fed to the RSC encoder 1. The resultant path through the trellis is shown in Fig. 6.16.

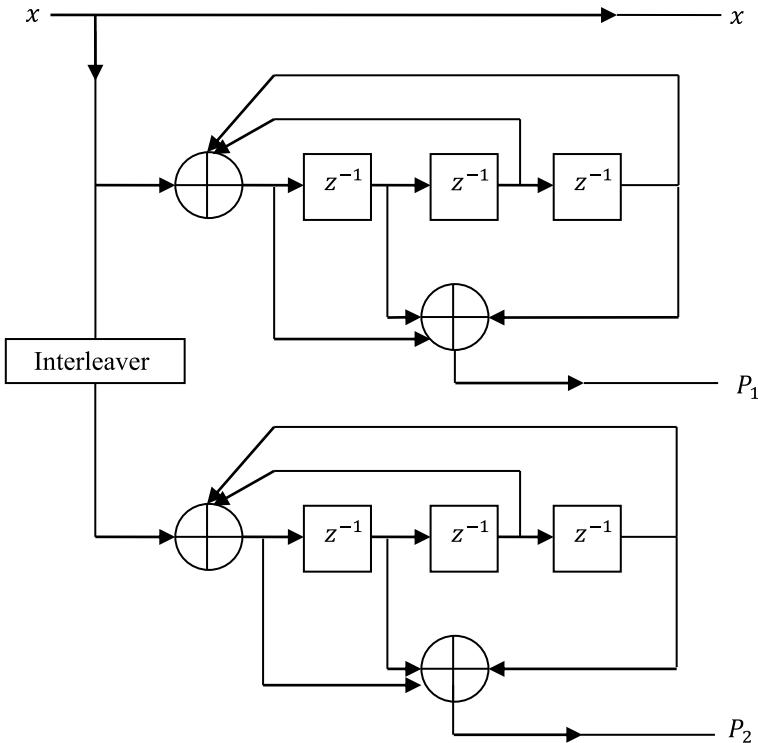


Fig. 6.14 Turbo encoder of Example 6.2

Now, the input is fed through the following pseudorandom interleaver shown in Fig. 6.17.

The block of permuted data bits is then fed into RSC encoder 2, resulting in the path through the trellis is shown in Fig. 6.18.

The encoder output data bits and the parity bits are mapped to symbols as shown in Table 6.4.

6.2.3 Turbo Coding Using MATLAB

Example 6.3 Consider the turbo encoder given in [1] using the RSC encoders with the generator function

$$G = \left[1 \quad \frac{1 + D^4}{1 + D + D^2 + D^3 + D^4} \right]$$

Table 6.3 Transitions for turbo encoder of Example 6.2

$x(n)$	State at n	State at $n + 1$	$P_1(n)$
0	S_0 (000)	S_0	0
1		S_1	1
0	S_1 (100)	S_2	1
1		S_3	0
0	S_2 (010)	S_5	1
1		S_4	0
0	S_3 (110)	S_7	0
1		S_6	1
0	S_4 (001)	S_1	0
1		S_0	1
0	S_5 (101)	S_3	1
1		S_2	0
0	S_6 (011)	S_4	1
1		S_5	0
0	S_7 (111)	S_6	0
1		S_7	1

- (a) Assuming RSC encoder 1 trellis is terminated determine the code word produced by the unpunctured encoder for the message $x = [1 \ 0 \ 0 \ 1 \ 1 \ 0]$ using MATLAB. Let the interleaver be $[3, 7, 6, 2, 5, 10, 1, 8, 9, 4]$.
- (b) Repeat (a) for punctured encoder with rate $\frac{1}{2}$
The puncturing patterns are $P_{u1} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}; P_{u2} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$
- (c) Assuming RSC encoder 1 trellis is unterminated determine the code word produced by the unpunctured encoder for the message $x = [1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0]$ using MATLAB. Let the interleaver be $[3, 7, 6, 2, 5, 10, 1, 8, 9, 4]$.
- (d) Repeat (c) for punctured encoder with rate $\frac{1}{2}$ with the puncturing patterns same as in (b).

Solution The following MATLAB program and MATLAB functions are written and used to find the codewords produced by the unpunctured and punctured encoders. For (a) and (b), the program is to be run with $ip = [1 \ 0 \ 0 \ 1 \ 1 \ 0]$ and term 1 = 1. Whereas for (c) and (d) with $ip = [1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0]$ and term 1 = -1.

- (a) The unpunctured turbo code obtained by running the MATLAB program and functions is

$$x = [1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0]; p_1 = [1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0]; p_2 = [0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1]$$
- (b) The punctured turbo code obtained is

$$x = [1 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0]; p_1 = [1 \ 0 \ 1 \ 0 \ 0]; p_2 = [1 \ 0 \ 0 \ 1 \ 1]$$

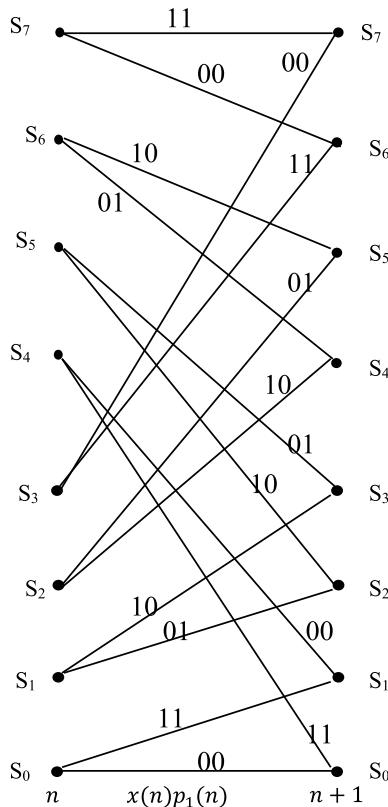


Fig. 6.15 Trellis diagram for encoder of Example 6.2

Since for every 10 information bits, there are 20 code word bits (10 information bits and five parity bits for each RSC encoder; thus, the rate of the punctured turbo code is $1/2$).

- (c) The unpunctured turbo code obtained is
 $x = [1\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 0]$;
 $p_1 = [1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 1]$; $p_2 = [0\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0]$
- (d) The punctured turbo code obtained is
 $x = [1\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 0]$; $p_1 = [1\ 0\ 1\ 0\ 0]$; $p_1 = [1\ 0\ 0\ 0\ 0]$

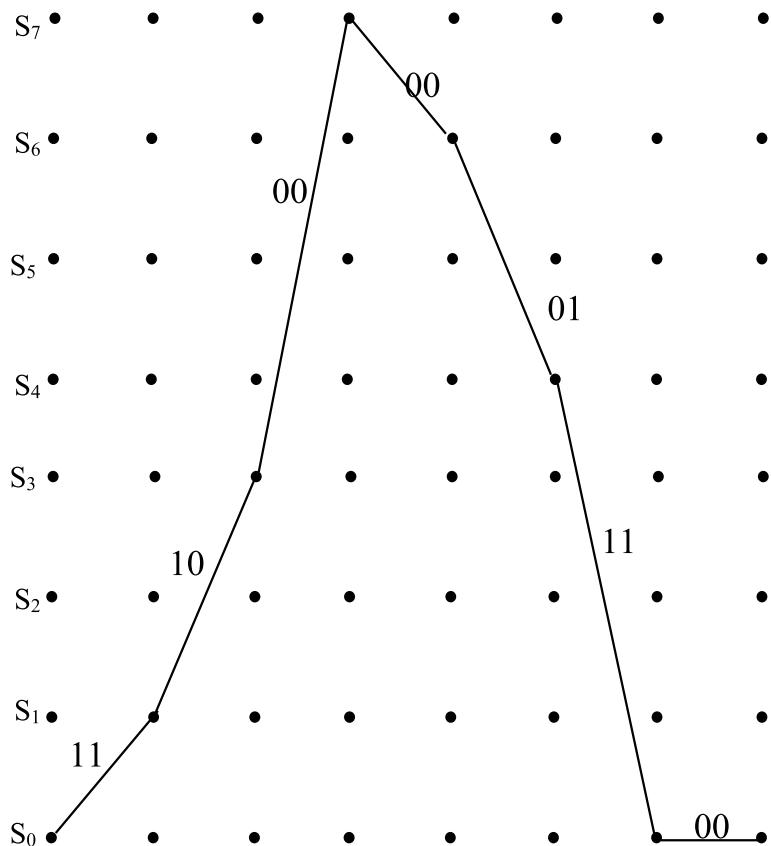


Fig. 6.16 Trellis path corresponding to input sequence Example 6.2

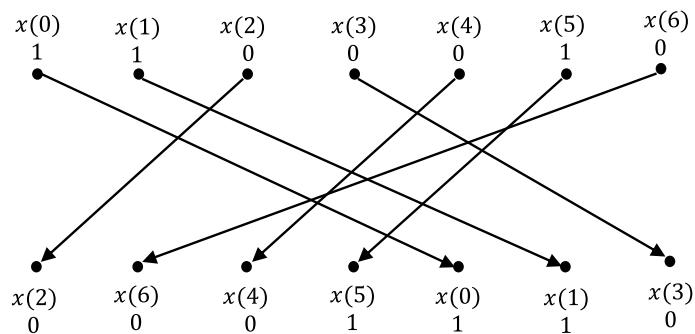


Fig. 6.17 Pseudorandom interleaver of Example 6.2

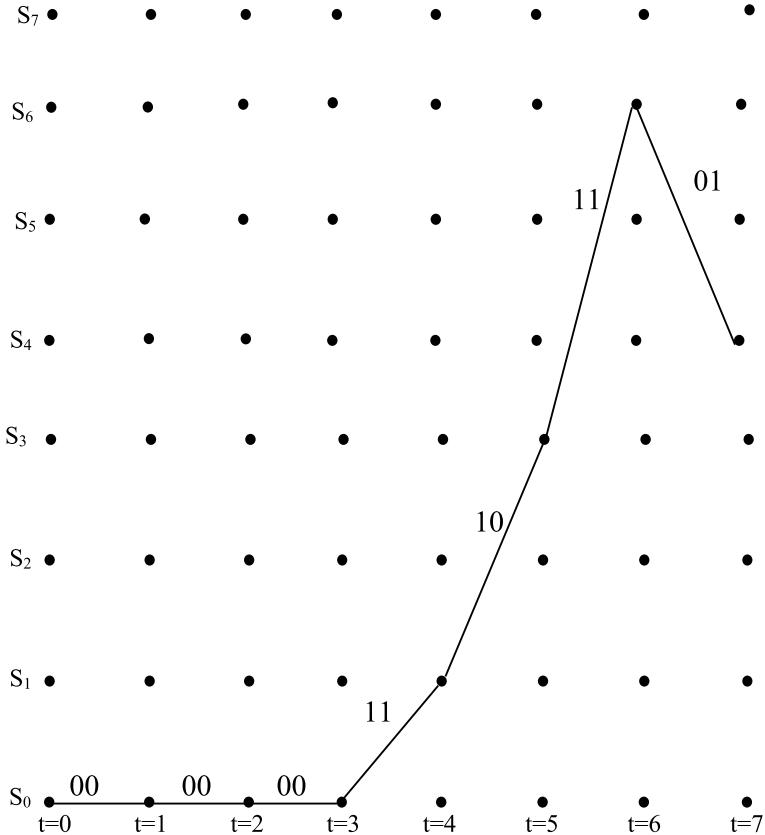


Fig. 6.18 Trellis path corresponding to interleaved input sequence of Example 6.2

Table 6.4 Output of the encoder of Example 6.2

$x(n)$	$p_1(n)$	$p_2(n)$	\rightarrow	$x(n)$	$p_1(n)$	$p_2(n)$
1	1	0		1	1	-1
1	0	0		1	-1	-1
0	0	0		-1	-1	-1
0	0	1		-1	-1	1
0	1	0		-1	1	-1
0	0	1		-1	-1	1

Program 6.1 MATLAB program to find output code word

```
clear all;
g = [ 1 1 1 1 1;1 0 0 0 1];
ip =[1 0 0 1 1 0 ]; % info. bits
[n,K] = size(g);
m = K - 1;
term1=input('enter 1 terminated trellises, -1 for unterminated trellises');
term2=-1;
if(term1==1)
    L=length(ip)+m;
else
    L=length(ip);
end
op1 = turboencode(g, ip,term1);
y(1,:) = op1(1:2:2*L);
y(2,:) = op1(2:2:2*L);
interl=[3 7 6 2 5 10 1 8 9 4];
for i = 1:L
    ip1(1,i) = y(1,interl(i));% interleaved input
end
op2= turboencode(g, ip1,term2);
y(3,:) = op2(2:2:2*L);
punc=input('enter 1 for unpunctured code, 0 for punctured code');
if(punc==1)
x=y(1,:);p1=y(2,:);p2=y(3,:);
y=[x' p1' p2'];
y= 2*y-ones(size(y));
end
if(punc==0)
x=y(1,:);p1=y(2,1:2:end);p2=y(3,2:2:end);
y11=x';
y12=[p1' p2'];
y1p= 2*y11-ones(size(y11));
y23p=2*y12-ones(size(y12));
end
```

MATLAB function encodedbit.m

```
function [output, state] = encodedbit(g, input, state)
[n,k] = size(g);
m = k-1;
for i=1:n
    output(i) = g(i,1)*input;
    for j = 2:k
        output(i) = xor(output(i),g(i,j)*state(j-1));
    end;
end
state = [input, state(1:m-1)];
```

MATLAB function turboencode.m

```
function y = turboencode(g, ip, terminated)
[n,K] = size(g);
m = K - 1;
if terminated>0
    Linf = length(ip);
    L = Linf + m;
else
    L = length(ip);
    Linf = L - m;
end
% initialize the state vector
state = zeros(1,m);
% generate the codeword
for i = 1:L
    if terminated<0 | (terminated>0 & i<=Linf )
        xk = ip(1,i);
    elseif terminated>0 & i>Linf
        % terminate the trellis
        xk = rem( g(1,2:K)*state', 2 );
    end
    xak = rem( g(1,:)*[xk state]', 2 );
    [outputbits, state] = encodedbit(g, xak, state);
    outputbits(1,1) = xk;
    y(n*(i-1)+1:n*i) = outputbits;
end
```

6.2.4 Encoding Tail-Biting Codes with RSC (Feedback) Encoders

For encoders with feedback, the situation is not like done in Chap. 5 for tail-biting codes with feed forward encoders. In encoding tail-biting codes with feedback encoders, the ending state depends on the entire information word. Thus, in order to fulfill the boundary condition for a given information word x , we have to calculate the initial state S_0 that will lead to the same state after N cycles. The correct initial state can be calculated using the state-space representation of the encoder.

A convolutional encoder with k inputs, n outputs, and m memory elements can be described by the following state-space equations over GF(2)

$$S_{j+1} = AS_j + Bx_j^T \quad (6.1a)$$

$$y_j = CS_j + Dx_j^T \quad (6.1b)$$

where S_j is the $(m \times 1)$ state vector, x_j is the $(1 \times k)$ input vector, where A is the $(m \times m)$ state matrix, B denotes the $(m \times k)$ control matrix, C is the $(n \times m)$ observation matrix, and D denotes the $(n \times k)$ transition matrix.

If we demand that the state at time N is equal to the initial state S_0 , we obtain from the solution of Eq. (6.1a)

$$[A^N + I_m]S_0 = S_N \quad (6.1c)$$

where I_m denotes the $(m \times m)$ identity matrix. Provided the matrix $[A^N + I_m]$ is invertible, the correct initial state can be calculated. The two-step encoding process is as follows:

Step 1: Start the encoder in the S_0 state, with all information bits as input and the output bits are ignored. After N cycles, the encoder is in the state S_N . We can calculate the corresponding initial state using Eq. (1c) to initialize the encoder for actual encoding.

Step 2: Start the encoder in the correct initial state as calculated in step 1; the information word x as input, and a valid codeword results.

Example 6.4 Consider the following systematic feedback encoder. Obtain tail-biting code for input $x = (1\ 100)$

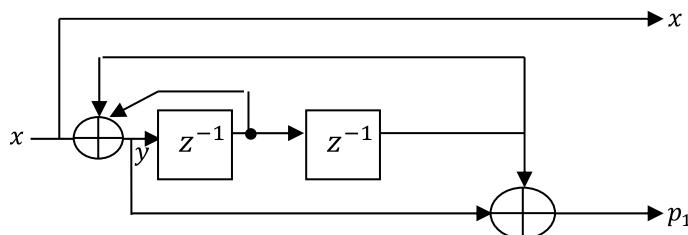


Table 6.5 Transitions for feedback encoder of Example 6.4

$x(n)$	State at $(n - 1)$	State at n	$P_1(n)$
0	S_0	S_0	0
1		S_1	1
0	S_1	S_3	1
1		S_2	0
0	S_2	S_1	0
1		S_0	1
0	S_3	S_2	1
1		S_3	0

Solution The two binary memory elements can assume any one of four states $S_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$; $S_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$; $S_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$; $S_3 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$; where the state $= \begin{bmatrix} y(n-2) \\ y(n-1) \end{bmatrix}$. The state transitions for the given feedback encoder are provided in Table 6.5.

For given encoder, the state-space representation is given by

$$S_{n+1} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} S_n + \begin{bmatrix} 0 \\ 1 \end{bmatrix} x(n)$$

$$P_1(n) = [01] S_n + x(n)$$

$$\text{where } S_n = \begin{bmatrix} y(n-2) \\ y(n-1) \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}; B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}; C = \begin{bmatrix} 0 & 1 \end{bmatrix}; D = 1.$$

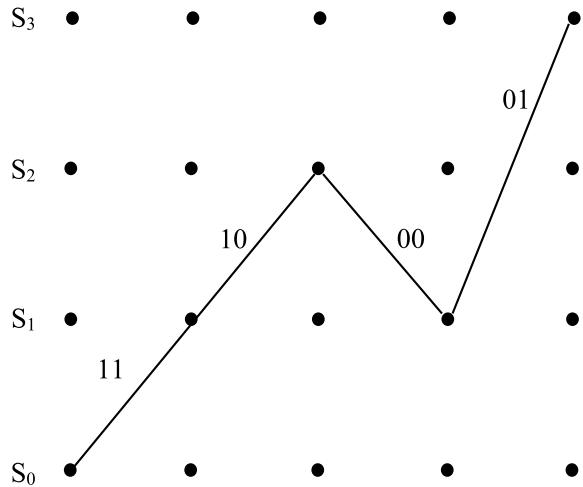
The number of cycles $N = 4$.

Step 1: The encoder is started in state S_0

After 4 cycles, the encoder is in the state S_3 as shown in Fig. 6.19. The corresponding initial state is given by solving

$$\begin{aligned} [A^4 + I_2] S_0 &= S_3 \\ \left[\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^4 + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right] S_0 &= S_3 \\ &= \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} S_0 = S_3 \\ S_0 &= \left[\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \right]^{-1} S_3 = \left[\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \right]^{-1} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \end{aligned}$$

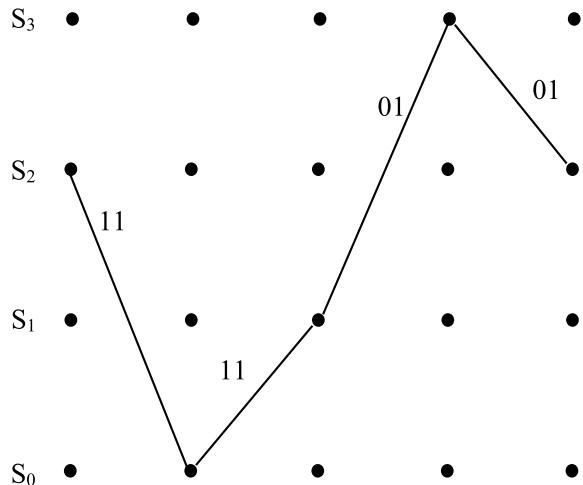
Fig. 6.19 Trellis path of tail-biting encoding step 1



$$= \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} y(n-2) \\ y(n-1) \end{bmatrix} = S_2.$$

Step 2: Now, the encoder is started in the correct initial state S_1 as shown in Fig. 6.20 and the valid code word is $[x(n)P_1(n)] = [11, 11, 01, 01]$.

Fig. 6.20 Trellis path of tail-biting encoding step 2 (actual encoding)



6.3 Turbo Decoder

The iterative decoding turbo decoder block diagram is shown in Fig. 6.21.

During the first iteration, no extrinsic information is available from decoder 2; hence, the a priori information is set to zero i.e., Then, the decoder outputs the estimate of the systematic bit as the log-likelihood ratio (LLR)

$$L_1(\hat{x}(n)) = \left(\frac{P(x(n) = 1)|x', p'_1, L_a(\hat{x})}{P(x(n) = 0)|x', p'_1, L_a(\hat{x})} \right) \quad n = 1, 2, \dots, N \quad (6.2a)$$

It is assumed that the message bits are statistically independent. Thus, the total LLR is given by

$$L_1(\hat{x}) = \sum_{n=1}^N L_1(\hat{x}(n)) \quad (6.2b)$$

Hence, the message bits' extrinsic information obtained from the first decoder is

$$L_{e1}(x) = L_1(x) - L_a(x) - L_c x' \quad (6.3)$$

The term $L_c x'$ is the information provided by the noisy observation. The extrinsic information $L_{e1}(x)$ and x' are interleaved before applying it as input to the BCJR algorithm in the second decoder. The noisy parity check bits p'_2 are also an additional input to the BCJR algorithm. The extrinsic information obtained from the BCJR algorithm is de-interleaved to produce the total log-likelihood ratio

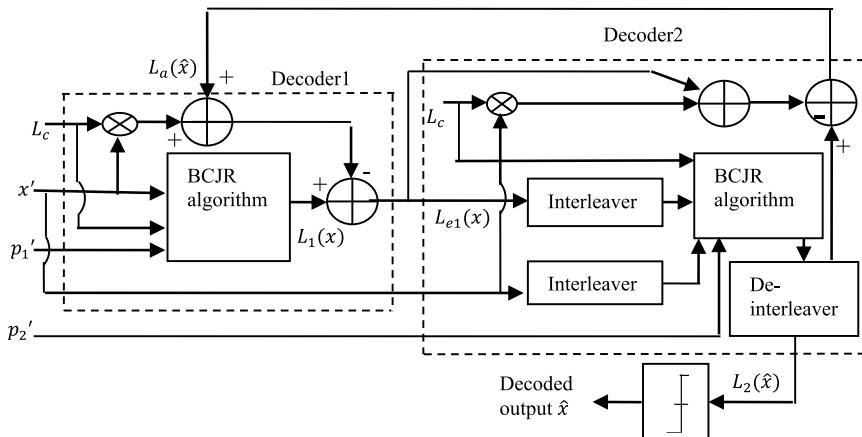


Fig. 6.21 Iterative decoding turbo decoder block diagram

$$L_2(x) = \sum_{n=1}^N \left(\frac{P(x(n) = 1)|x', p'_2, L_{e1}(x)}{P(x(n) = 0)|x', p'_2, L_{e1}(x)} \right) \quad (6.4)$$

is hard limited to estimating the information bit based only on the sign of the de-interleaved LLR, at the output of the decoder as expressed by

$$\hat{x} = \text{sgn} (L_2(x)) \quad (6.5)$$

The extrinsic information

$$L_a(x) = L_2(x) - (L_{e1}(x) + L_c x') \quad (6.6)$$

is fed back to the decoder 1. The extrinsic information of one decoder is used as the a priori input to the other decoder, and thus in the turbo decoder iterations, the extrinsic information ping-ponged back and forth between MAP decoders.

After a certain number of iterations, the log-likelihood $L_2(x)$ at the output of decoder 2 is de-interleaved and delivered to the hard decision device, which estimates the input.

If it is assumed that $x(n) = \pm 1$ is transmitted over a Gaussian or fading channel using BPSK modulation, the probability of the matched filter output $y(n)$ is given by Hanzo et al. [4]

$$P(y(n)|x(n) = +1) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{E_b}{2\sigma^2}(y(n) - a)^2\right) \quad (6.7a)$$

where E_b is the transmitted energy per bit, σ^2 is the noise variance, and a is the fading amplitude. Similarly,

$$P(y(n)|x(n) = -1) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{E_b}{2\sigma^2}(y(n) + a)^2\right) \quad (6.7b)$$

Therefore, when we use BPSK over a (possibly fading) Gaussian channel, the term $L_c x'(n)$ can be expressed as

$$\begin{aligned} L_c(x'(n)|x(n)) &= \log\left(\frac{P(x'(n)|x(n) = +1)}{P(x'(n)|x(n) = -1)}\right) \\ &= \log\left(\frac{\exp\left(-\frac{E_b}{2\sigma^2}(x'(n) - a)^2\right)}{\exp\left(-\frac{E_b}{2\sigma^2}(x'(n) + a)^2\right)}\right) \\ &= \left(-\frac{E_b}{2\sigma^2}(x'(n) - a)^2\right) - \left(-\frac{E_b}{2\sigma^2}(x'(n) + a)^2\right) \\ &= \frac{E_b}{2\sigma^2} 4a \cdot x'(n) \end{aligned}$$

$$= L_c x'(n) \quad (6.8)$$

where

$$L_c = 4a \frac{E_b}{2\sigma^2}$$

is defined as the channel reliability value.

6.3.1 The BCJR Algorithm

The BCJR algorithm was published in 1974. It is named after its inventors: Bahl, Cocke, Jelinek, and Raviv. It is for maximum a posteriori (MAP) decoding of codes defined on trellises [5]. It was not used in practical implementations for about 20 years due to more complexity than the Viterbi algorithm. The BCJR algorithm was reborn vigorously when the turbo code inventors Berrou, Glavieux, and Thitimajshima [1] used a modified version of the BCJR algorithm in 1993. Consider a trellis section with four states like the one presented in Fig. 6.22.

In the trellis section, the branches generated by input message bits 1 and -1 are represented by a dashed line and a solid line, respectively. The variable $\gamma(n)$ represents the branch probabilities at time n , and the variables $\alpha(n)$ and $\beta(n)$ are the forward and backward estimates of the state probabilities at time n based on the past and future data, respectively. Now the log-likelihood ratios expressed by Eqs. (6.2a),

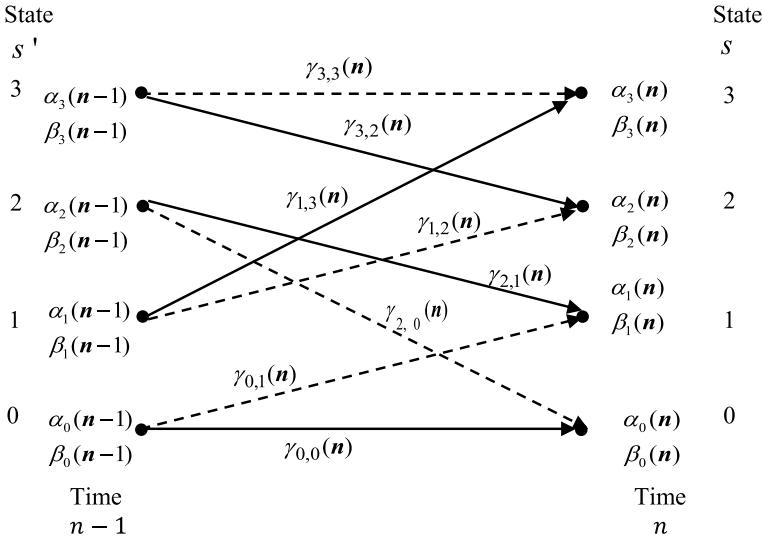


Fig. 6.22 Typical trellis section with α , β , γ as labels

6.2b) and (6.4) can be computed, using branch probabilities, forward, and backward error probabilities of the states as

$$L_1(\hat{x}) = \log \left[\frac{\sum_{R_1} \alpha_{s'}(n-1) \cdot \gamma_{s',s}(n) \cdot \beta_s(n)}{\sum_{R_0} \alpha_{k-1}(s') \cdot \gamma_k(s', s) \cdot \beta_k(s)} \right] \quad (6.9)$$

where s represents the state at time n and s' stands for the previous state, i.e., the state at time instant $n - 1$ as in a typical trellis section shown in Fig. 6.22. The R_1 indicates the summation computed over all the state transitions from s' to s due to message bits $x(n) = +1$ (i.e., dashed branches). The denominator R_0 is the set of all branches originated by message bits $x(n) = -1$.

For a given state transition, the transmitted signal is the data bit and parity check bit pair. Also, for a given starting state, the data bit value determines the next state. Using the Bayes' theorem, the branch probability can be expressed as [7]

$$\gamma_{s',s} = \Pr(x(n)) \Pr(y'(n)|y(n)) \quad (6.10)$$

The probability of the data bit $x(n)$ in terms of the a priori probability ratio can be written as follows:

$$\begin{aligned} \Pr(x(n)) &= \frac{\exp\left[\frac{1}{2}L_a(x(n))\right]}{1 + \exp[L_a(x(n))]} \cdot \exp\left[\frac{1}{2}x_k L_a(x(n))\right] \\ &= B_n \cdot \exp\left[\frac{1}{2}x(n)L_a(x(n))\right] \end{aligned} \quad (6.11)$$

$$L_a(x(n)) = \log \left[\frac{\Pr(x(n) = +1)}{\Pr(x(n) = -1)} \right] \quad (6.12)$$

The probability of the noisy data bit $x'(n)$ and parity bits $p'(n)$ can be expressed in terms of Gaussian probability distributions as

$$\begin{aligned} \Pr(y'(n)|y(n)) &= \Pr(x'(n)|x(n)) \cdot \Pr(p'(n)|p(n)) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left[-\frac{(x'(n) - x(n))^2}{2\sigma^2}\right] \cdot \Delta \\ &\quad \cdot \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left[-\frac{(p'(n) - p(n))^2}{2\sigma^2}\right] \cdot \Delta \\ &= A_n \cdot \exp\left[\frac{x'(n)x(n) - p'(n)p(n)}{\sigma^2}\right] \end{aligned} \quad (6.13)$$

Since $\gamma_{s',s}$ appears in the numerator (where $x(n) = +1$), and denominator (where $x(n) = -1$) of Eq. (6.9), the $A_n B_n$ factor will get canceled as it is independent of $x(n)$. Thus, the branch probability $\gamma_{s',s}(n)$ can be expressed as

$$\gamma_{s',s}(n) = \exp\left[\frac{1}{2}\left(x(n)L_a x(n) + x(n)L_c x'(n) + p(n)L_c p'(n)\right)\right] \quad (6.14)$$

The forward recursion is computed as

$$\alpha_s(n) = \sum_{s'} \alpha_{s'}(n-1) \gamma_{s',s}(n) \quad (6.15)$$

The backward recursion is computed as

$$\beta_{s'}(n-1) = \sum_{s'} \gamma_{s',s}(n) \cdot \beta_s(n) \quad (6.16)$$

$$\alpha_0(n) = \alpha_0(n-1) \gamma_{0,0}(n) + \alpha_0(n-1) \gamma_{2,0}(n) \quad (6.17a)$$

$$\beta_0(n) = \beta_1(n+1) \gamma_{0,1}(n+1) + \beta_0(n+1) \gamma_{2,0}(n+1) \quad (6.17b)$$

The recursive calculation of $\alpha_0(n)$ and $\beta_0(n)$ as in Eqs. (6.17a) and (6.17b) is illustrated in Fig. 6.23.

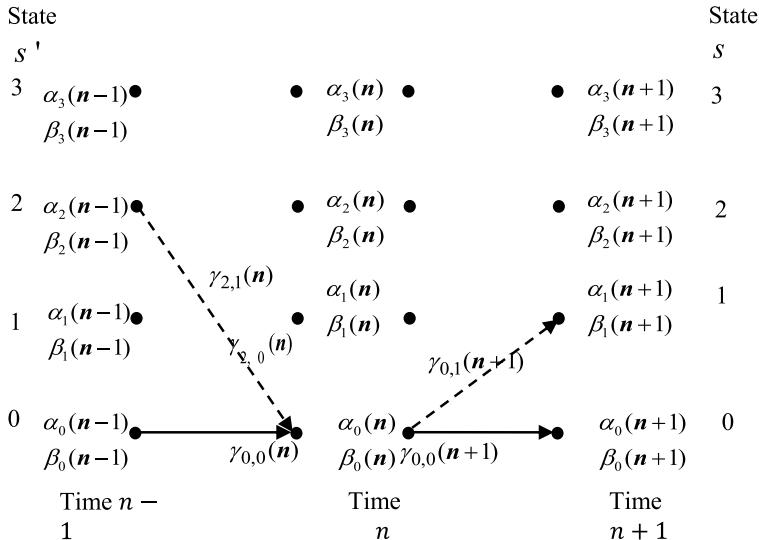


Fig. 6.23 Illustration of recursive calculation of $\alpha_0(n)$ and $\beta_0(n)$

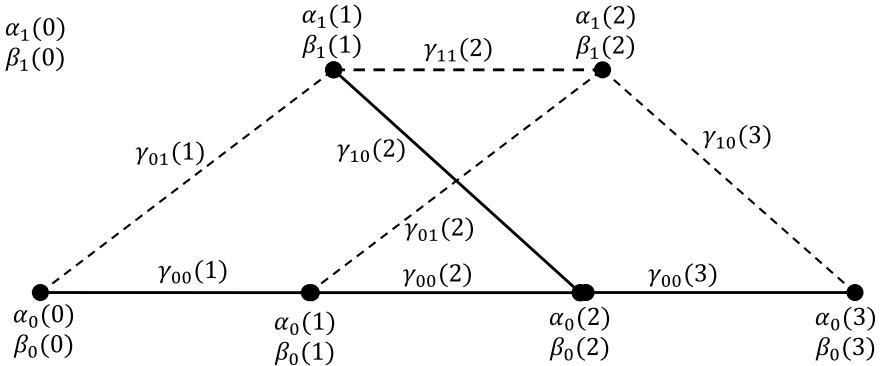


Fig. 6.24 Trellis diagram of Example 6.4

The following simple example illustrates the recursive computation of forward and backward state error probabilities.

Example 6.4 Consider the following trellis diagram shown in Fig. 6.24 with the following probabilities.

$$\begin{aligned}\gamma_{00}(1) &= 0.48, \gamma_{00}(2) = 0.485, \gamma_{00}(3) = 0.6, \gamma_{01}(1) = 0.09, \\ \gamma_{01}(2) &= 0.1, \gamma_{10}(2) = 0.1, \gamma_{10}(3) = 0.7, \gamma_{11}(2) = 0.485\end{aligned}$$

Compute the forward and backward state error probabilities of the trellis.

Solution With the initial value $\alpha_0(0) = 1$, the forward recursion yields the values

$$\begin{aligned}\alpha_0(1) &= \alpha_0(0)\gamma_{00}(1) = 0.48 \\ \alpha_1(1) &= \alpha_0(0)\gamma_{01}(1) = 0.09 \\ \alpha_0(2) &= \alpha_0(1)\gamma_{00}(2) + \alpha_1(1)\gamma_{10}(2) = 0.2418 \\ \alpha_1(2) &= \alpha_0(1)\gamma_{01}(2) + \alpha_1(1)\gamma_{11}(2) = 0.0916 \\ \alpha_0(3) &= \alpha_0(2)\gamma_{00}(3) + \alpha_1(2)\gamma_{10}(3) = 0.2092\end{aligned}$$

With the initial value $\beta_0(3) = 1$, the backward recursion yields the values

$$\begin{aligned}\beta_0(2) &= \beta_0(3)\gamma_{00}(3) = 0.6 \\ \beta_1(2) &= \beta_0(3)\gamma_{10}(3) = 0.7 \\ \beta_0(1) &= \beta_0(2)\gamma_{00}(2) + \beta_1(2)\gamma_{01}(2) = 0.3510 \\ \beta_1(1) &= \beta_0(2)\gamma_{10}(2) + \beta_1(2)\gamma_{11}(2) = 0.3995 \\ \beta_0(0) &= \beta_0(1)\gamma_{00}(1) + \beta_1(1)\gamma_{01}(1) = 0.2044\end{aligned}$$

6.3.2 Turbo Decoding Illustration

Example 6.5 Assume the channel adds unity variance Gaussian noise to the code generated by the turbo encoder considered in Example 6.1, decode the received sequence.

Solution For a random run with unity variance Gaussian noise, the received code is

$x'(n)$	$p'_1(n)$	$p'_2(n)$
0.9039	0.1635	1.2959
-0.3429	-0.2821	2.6317
-1.6952	-0.1377	2.2603
-2.6017	-0.4482	1.5740
-1.4019	1.6934	1.8197
1.7155	2.2967	1.6719

Using the trellis diagram shown in Fig. 6.1, the branch probabilities for the first stage are computed as

$$\begin{aligned}\gamma_{0,0}(1) &= \exp(-x'(0) - p'_1(0)) = \exp(-0.9039 - 0.1635) = 0.34390149993822 \\ \gamma_{0,1}(1) &= \exp(x'(0) + p'_1(0)) = \exp(0.9039 + 0.1635) = 2.90780935872520 \\ \gamma_{1,2}(1) &= \exp(x'(0) - p'_1(0)) = \exp(0.9039 - 0.1635) = 2.09677405639736 \\ \gamma_{1,3}(1) &= \exp(-x'(0) + p'_1(0)) = \exp(-0.9039 + 0.1635) = 0.47692310811885 \\ \gamma_{2,0}(1) &= \exp(x'(0) + p'_1(0)) = \exp(0.9039 + 0.1635) = 2.90780935872520 \\ \gamma_{2,1}(1) &= \exp(-x'(0) - p'_1(0)) = \exp(-0.9039 - 0.1635) = 0.34390149993822 \\ \gamma_{3,2}(1) &= \exp(-x'(0) + p'_1(0)) = \exp(-0.9039 + 0.1635) = 0.47692310811885 \\ \gamma_{3,3}(1) &= \exp(x'(0) - p'_1(0)) = \exp(0.9039 - 0.1635) = 2.09677405639736\end{aligned}$$

Repeating the branch probabilities computation for other stages of the trellis, the branch probabilities for all stages of the trellis for this example are given as follows:

n	$\gamma_{0,0}(n)/\gamma_{2,1}(n)$	$\gamma_{0,1}(n)/\gamma_{2,0}(n)$	$\gamma_{1,2}(n)/\gamma_{3,3}(n)$	$\gamma_{1,3}(n)/\gamma_{3,2}(n)$
1	0.34390149993822	2.90780935872520	2.09677405639736	0.47692310811885
2	1.86824595743222	0.53526142851899	0.94101142324168	1.06268635566092
3	6.25199116868593	0.15994904231610	0.21066206860156	4.74693905095638
4	21.11323299367156	0.04736366052038	0.11607717585511	8.61495804522538
5	0.74714201360297	1.33843363349046	0.04526143199369	22.0938657031320
6	0.01809354561793	55.26832723205136	0.55922689149115	1.78818296332918

The forward state probabilities are computed using Eq. (6.15), and the resulting normalized forward state probabilities are given as follows:

n	$\alpha_0(n)$	$\alpha_1(n)$	$\alpha_2(n)$	$\alpha_3(n)$
0	1.000000000000000	0	0	0
1	0.10576017207126	0.89423982792874	0	0
2	0.09657271753492	0.02766854681958	0.41128905912021	0.46446967652529
3	0.11754442548550	0.45413087700815	0.38808965080132	0.04023504670503
4	0.16649906702111	0.54604949205154	0.02659440531342	0.26085703561394
5	0.00875863686136	0.01328728572633	0.31685999430997	0.66109408310235
6	0.89416277694224	0.02500892891120	0.06073869015360	0.02008960399296

The backward state probabilities are computed using Eq. (6.16), and the resulting normalized backward probabilities for this example are given as follows:

n	$\beta_0(n)$	$\beta_1(n)$	$\beta_2(n)$	$\beta_3(n)$
6	0.45530056303648	0.09512620145755	0.07905302691388	0.37052020859209
5	0.02546276878415	0.45050410997512	0.01580448043695	0.50822864080378
4	0.02462941844490	0.01001105352469	0.96125810870846	0.00410141932195
3	0.00003769540412	0.98178511456649	0.00492923752535	0.01324795250404
2	0.00001104782103	0.00204434600799	0.00001979111732	0.99792481505365
1	0.00032726925280	0	0.99967273074720	0
0	1.000000000000000	0	0	0

Now, using Eq. (6.9), we compute the $L_1(x)$ the LLR from the decoder 1 as

$$L_1(x(n)) = \frac{\alpha_0(n-1)\beta_1(n)\gamma_{0,1}(n) + \alpha_1(n-1)\beta_2(n)\gamma_{1,2}(n) + \alpha_2(n-1)\beta_0(n)\gamma_{2,0}(n) + \alpha_3(n-1)\beta_3(n)\gamma_{3,3}(n)}{\alpha_0(n-1)\beta_0(n)\gamma_{0,0}(n) + \alpha_1(n-1)\beta_3(n)\gamma_{1,3}(n) + \alpha_2(n-1)\beta_1(n)\gamma_{2,1}(n) + \alpha_3(n-1)\beta_2(n)\gamma_{3,2}(n)} \quad (6.18)$$

The resulting LLR from the decoder 1 is

$$L_1(x) = \begin{bmatrix} 5.00794986257243 \\ 4.52571043846903 \\ -5.03587871714769 \\ -6.73223129201010 \\ -5.45139857004191 \\ 11.61281973458293 \end{bmatrix}$$

and

$$L_c(x) = \begin{bmatrix} 1.807800000000000 \\ -0.685800000000000 \\ -3.390400000000000 \\ -5.203400000000000 \\ -2.803800000000000 \\ 3.431000000000000 \end{bmatrix}$$

The soft and hard decisions are:

$L_1(\hat{x}(n))$	$\hat{x}(n) = \text{sign}(L_1(\hat{x}(n)))$
5.0079	1
4.5257	1
-5.035	0
-6.732	0
-5.451	0
11.61	1

When compared with the trellis path of decoder 1 shown in Fig. 6.11, it is observed that the decoder has correctly estimated all data bits. Since a priori information for the first iteration is zero, the extrinsic information $L_{e1}(x)$ is given

$$L_{e1}(x) = L_1(x) - L_c(x) = \begin{bmatrix} 5.00794986257243 \\ 4.52571043846903 \\ -5.03587871714769 \\ -6.73223129201010 \\ -5.45139857004191 \\ 11.61281973458293 \end{bmatrix} - \begin{bmatrix} 1.8078 \\ -0.6858 \\ -3.3904 \\ -5.2034 \\ -2.8038 \\ 3.4310 \end{bmatrix} = \begin{bmatrix} 3.20014986257242 \\ 5.21151043846903 \\ -1.64547871714763 \\ -1.52883129201010 \\ -2.64759857004191 \\ 8.18181973458293 \end{bmatrix}$$

The extrinsic information $L_{e1}(x)$ from the decoder 1 and the noisy information bits $x'(n)$ are to be interleaved before feeding as inputs to the BCJR algorithm of the decoder 2. After relabeling, the following are the inputs to BCJR of the decoder 2.

$L_{e1}(x)$	$x'(n)$	$p'_2(n)$
8.18181973458293	1.7155	1.2959
-1.64547871714769	-1.6952	2.6317
-2.64759857004191	-1.4019	2.2603
3.20014986257242	0.9039	1.574
5.21151043846903	-0.3429	1.8197
-1.52883129201010	-2.6017	1.6719

Using the trellis diagram shown in Fig. 6.1, the branch probabilities for the first stage are computed as

$$\begin{aligned}
\gamma_{0,0}(1) &= \exp(-0.5 * L_{e1}(x) - x'(0) - p'_1(01)) \\
&= \exp(-0.5 * 8.18181973458293 - 1.7155 - 1.2959) = 0.00082320123987 \\
\gamma_{0,1}(1) &= \exp(0.5 * L_{e1}(x) + x'(0) + p'_1(01)) \\
&= \exp(0.5 * 8.18181973458293 + 1.7155 + 1.2959) = 1214.7697933.438 \\
\gamma_{1,2}(1) &= \exp(0.5 * L_{e1}(x) + x'(0) - p'_1(01)) \\
&= \exp(0.5 * 8.18181973458293 + 1.7155 - 1.2959) = 90.9681883921071 \\
\gamma_{1,3}(1) &= \exp(-0.5 * L_{e1}(x) - x'(0) + p'_1(01)) \\
&= \exp(-0.5 * 8.18181973458293 - 1.7155 + 1.2959) = 0.01099285385007 \\
\gamma_{2,0}(1) &= \exp(0.5 * L_{e1}(x) + x'(0) + p'_1(01)) \\
&= \exp(0.5 * 8.18181973458293 + 1.7155 + 1.2959) = 1214.7697933.438 \\
\gamma_{2,1}(1) &= \exp(-0.5 * L_{e1}(x) - x'(0) - p'_1(01)) \\
&= \exp(-0.5 * 8.18181973458293 - 1.7155 - 1.2959) = 0.00082320123987 \\
\gamma_{3,2}(1) &= \exp(-0.5 * L_{e1}(x) - x'(0) + p'_1(01)) \\
&= \exp(-0.5 * 8.18181973458293 - 1.7155 + 1.2959) = 0.01099285385007 \\
\gamma_{3,3}(1) &= \exp(0.5 * L_{e1}(x) + x'(0) - p'_1(01)) \\
&= \exp(0.5 * 8.18181973458293 + 1.7155 - 1.2959) = 90.9681883921071
\end{aligned}$$

Repeating the branch probabilities computation for other stages of the trellis, the branch probabilities for all stages of the trellis for this example are

N	$\gamma_{0,0}(n)/\gamma_{2,1}(n)$	$\gamma_{0,1}(n)/\gamma_{2,0}(n)$	$\gamma_{1,2}(n)/\gamma_{3,3}(n)$	$\gamma_{1,3}(n)/\gamma_{3,2}(n)$
1	0.00082320123987	1214.76979330438	90.96818839210695	0.010992853850
2	0.89247155103612	1.12048389536	0.00580149660962	172.369315590337
3	1.59264998322900	0.62788435032	0.00683294655359	146.349747090304
4	0.01694173912378	59.02581740243	2.53444564152908	0.394563601450
5	0.01686431851993	59.29679274132	1.55761408724524	0.642007547433
6	5.44237554167172	0.18374329231	0.00648660728077	154.163795758670

The forward recursion can be calculated using Eq. (6.15). The resulting normalized values are as follows:

n	$\alpha_0(n)$	$\alpha_1(n)$	$\alpha_2(n)$	$\alpha_3(n)$
0	1.000000000000000	0	0	0
1	0.00000067765982	0.99999932234018	0	0
2	0.00000000350858	0.00000000440497	0.00003365622987	0.99996633585658
3	0.00000014443156	0.00000036627375	0.99995279793500	0.00004669135969
4	0.99971058159622	0.00028708385150	0.00000032776038	0.00000200679189
5	0.00028464899198	0.99970462721982	0.00000756282988	0.00000316095832
6	0.00001006025926	0.00000060639718	0.00004523543737	0.99994409790619

The backward recursion can be calculated according to Eq. (6.16). The resulting normalized values are as follows:

n	$\beta_0(n)$	$\beta_1(n)$	$\beta_2(n)$	$\beta_3(n)$
6	0.99998080264562	0.00000062587196	0.00001074035175	0.00000783113066
5	0.00001006166024	0.99987703069337	0.00000834425763	0.00010456338877
4	0.00013868974158	0.00143501008710	0.00007080225952	0.99835549791180
3	0.01189148350941	0.00173100215279	0.98500767914419	0.00136983519361
2	0.93003760467044	0.01096095473547	0.03420391063551	0.02479752995857
1	0.01760402234259	0.48239597765741	0.01760402234259	0.48239597765741
0	0.2500000000000000	0.2500000000000000	0.2500000000000000	0.2500000000000000

Now, we compute the LLR from the decoder 2 using Eq. (6.18). The resulting LLR from the decoder 2 is

$$L_2(x) = \begin{bmatrix} 25.71127513129983 \\ -19.85060333367479 \\ -16.52345972282743 \\ 12.59341638166602 \\ 11.21364990579842 \\ -10.0677959239041 \end{bmatrix}$$

By slicing the soft decisions, we get the hard decisions 100110. Comparing this with the encoder 2 trellis path in Fig. 6.13, it is observed that the decoder has correctly estimated all data bits.

6.3.2.1 Turbo Decoding Using MATLAB

The following example illustrates the turbo decoding using MATLAB.

Example 6.6 When the code generated by the turbo encoder shown in Fig. 6.9 is transmitted over an AWGN channel with channel reliability factor $L_2 = 2$, sequence is received. Decode the received sequence using MATLAB.

n	$x'(n)$	$p'_1(n)$	$p'_2(n)$
1	3.01	3.13	-1.7
2	-0.23	-1.45	-1.7
3	-0.25	-0.18	1.82
4	0.83	0.91	2.0
5	-0.26	-0.45	-3.0
6	-0.8	1.3	1.46

(continued)

(continued)

n	$x'(n)$	$p'_1(n)$	$p'_2(n)$
7	0.43	1.98	2.1
8	-0.74	-0.54	0.3

Solution The following MATLAB program and MATLAB functions are written and used to decode the received sequence. After the first iteration, output of the decoder becomes as follows:

$L_1(\hat{x}(n))$	$\hat{x}(n) = \text{sign}(L_1(\hat{x}(n)))$	$L_1(\hat{x}(n))$	$\hat{x}(n) = \text{sign}(L_1(\hat{x}(n)))$
11.3900	1	11.9775	1
3.7217	1	7.1019	1
0.3753	1	-5.1767	0
0.4850	1	-4.5544	0
-0.4167	0	4.8071	1
-4.4225	0	-11.0942	0
3.7418	1	5.7408	1
-3.8245	0	-11.3693	0

After the second iteration, output of the decoder becomes as follows:

$L_1(\hat{x}(n))$	$\hat{x}(n) = \text{sign}(L_1(\hat{x}(n)))$	$L_1(\hat{x}(n))$	$\hat{x}(n) = \text{sign}(L_1(\hat{x}(n)))$
20.9509	1	20.9633	1
13.5723	1	28.2036	1
-15.2116	0	-23.1493	0
-14.6617	0	-17.3413	0
13.4480	1	21.2953	1
-19.1687	1	-30.6052	0
15.5069	0	18.3074	1
-21.0533	1	-33.4231	0

Thus, the transmitted input sequence $x(n) = (11001010)$.

Program 6.2 MATLAB program to decode the received sequence of Example 6.6

```
clear all;
g = [ 1 1 1;
       1 0 1 ];
[n,K] = size(g); m = K - 1;[n,K] = size(g);
m = K-1; % number of memories
nstates=2^m;% number of states
length=8;
r=[ 3.01      3.13      -1.7
    -0.23     -1.45      -1.7
    -0.25     -0.18      1.82
    0.83      0.91      2.0
   -0.26     -0.45      -3.0
   -0.8       1.3       1.46
    0.43      1.98      2.1
   -0.74     -0.54      0.3];
decin=[r(:,1)'; r(:,2)'; r(:,3)'];
xdec1=decin(1,:);
p1=decin(2,:);
interl=[ 8 6 2 3 5 4 7 1];
xdec2=xdec1(1,interl);
p2=decin(3,:);
La1=zeros(length,1);
for i=1:2
[L1 Le1] = turbodec(g, xdec1,p1,2, length,La1,nstates,1);
La2=Le1(interl');
[L2 Le2] = turbodec(g, xdec2,p2,2, length, La2, nstates,2);
deinterl=[8 3 4 6 5 2 7 1];
La1=Le2(deinterl');
L2=L2(deinterl');
end
```

MATLAB function trellis.m

```
function [trellismat, paritymat] = trellis(g);
[n,K] = size(g);
m = K - 1;
nstates = 2^m;
trellismat= zeros(nstates);
paritymat= zeros(nstates);
for s=1: nstates
decs=s-1; i=1;
% decimal to binary state
while decs >=0 & i<=m
bins(i) = rem( decs,2) ;
decs = (decs- bins(i))/2;
i=i+1;
end
bins=bins(m:-1:1);
% next state when input is 0
a = rem( g(1,:)*[0 bins ]', 2 );
v = g(2,1)*a;
for j = 1:K-1
v = xor(v, g(2,j+1)*bins(j));
end;
nstate0 = [a bins(1:m-1)];
% binary to decimal state
d=2.^{m-1:-1:0};
j=nstate0*d'+1;
trellismat(s,j)= -1;
paritymat(s,j)= 2*v-1;
% next state when input is 1
a = rem( g(1,:)*[1 bins]', 2 );
v = g(2,1)*a;
for j = 1:K-1
v = xor(v, g(2,j+1)*bins(j));
end;
nstate1 = [a bins(1:m-1)];
d=2.^{m-1:-1:0};
j=nstate1*d'+1;
trellismat(s,j)= 1;
paritymat(s,j)= 2*v-1;
end
```

MATLAB function turbodec.m

```

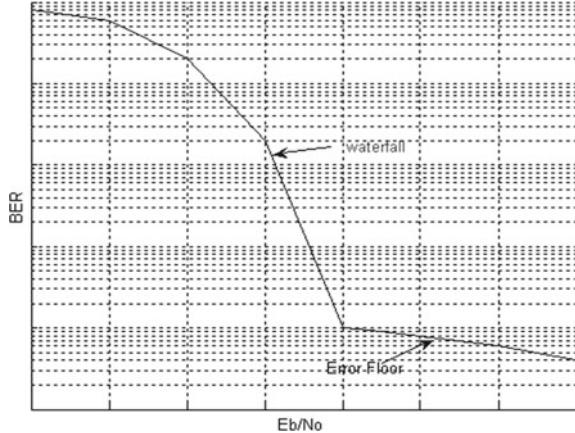
function [LMAP LE] = turbodec(g, x,p,Lc, bs, La,nst,ind)
[trellismat, paritymat]= trellis(g);
LMAP=zeros(bs,1);
LE=zeros(bs,1);
%Initialization Decoder
alpha= zeros(nst,bs+1);alpha(1,1)=1;
beta= zeros(nst,bs+1);
if (ind==1)
beta(1,end)=1;
elseif(ind==2)
beta(:,end)=1/(nst)*ones(nst,1);
end
gamma(1,1:bs)= struct('gammamat',zeros(size(trellismat)));
for k=1:bs
    gamma(k).gammamat= exp(1/2*(trellismat.*(La(k) + Lc*x(k)') +
Lc*paritymat.*p(k)'));
    gamma(k).gammamat= gamma(k).gammamat.*(trellismat>0 | trellismat<0);
end
%Compute alpha1
for k=1:bs
    alpha(:,k+1)= (gamma(k).gammamat)*(alpha(:,k));
    alpha(:,k+1)=alpha(:,k+1)/sum(alpha(:,k+1));
end
%Compute beta1 with formula (15)
for k=bs:-1:1
    beta(:,k)=(gamma(k).gammamat)'*(beta(:,k+1));
    beta(:,k)=beta(:,k)/sum(beta(:,k));
end
for k=1:bs
sumSplus=alpha(:,k)'*(gamma(k).gammamat.*(trellismat>0))*beta(:,k+1);
sumSminus=alpha(:,k)'*(gamma(k).gammamat.*(trellismat<0))*beta(:,k+1);
LMAP(k)=log(sumSplus/sumSminus);
end
LE=LMAP-La-Lc*x';
end

```

6.3.3 Convergence Behavior of the Turbo Codes

A typical BER curve for a turbo code is shown in Fig. 6.25. Three regions, namely low E_b/N_0 region, waterfall region, and error floor region, can be identified. In

Fig. 6.25 Typical BER curve of turbo codes



the low E_b/N_0 region, BER decreases slowly as E_b/N_0 increases. For intermediate values of E_b/N_0 , the BER decreases rapidly in the waterfall region with an increase in E_b/N_0 . In this region, the coding gain approaches the theoretical limit. For large E_b/N_0 , error floor occurs where the performance is dependent on the minimum Hamming distance of the code. The error floor is due to the weight distribution of turbo codes. Normally, turbo codes do not have large minimum distances. Hence, lowering the error floor results in better codes, which in some cases may result in faster convergence in decoding. One effective way of lowering the error floor is to use appropriate interleaver.

6.3.4 EXIT Analysis of Turbo Codes

Extrinsic information transfer (EXIT) chart [7] can be used as a tool to aid the construction of turbo codes. An EXIT chart is the reunion of two curves that characterize the two decoders used in a turbo decoder. Each curve represents a relation between the input and the output of one decoder. This relation is the mutual information between the output of the decoder (Le: the extrinsic information) and the initial message that was encoded before passing through the channel, with respect to the mutual information between the input of the decoder (La: the a priori information) and the message:

In a turbo decoder, the extrinsic information of the first decoder (Le1) is used as the a priori information of the second decoder (La2) and vice versa. It is suggested in [6] that a priori input to the constituent decoder can be modeled by

$$L_a = \mu_a \cdot x + \eta_a \quad (6.19)$$

where x is the known transmitted systematic bits, η_a is the Gaussian noise, and $\mu_a = \frac{\sigma_a^2}{2}$.

For each L_a , the mutual information I_A and I_E are computed as [6]

$$I_A = \frac{1}{2} \int \sum_{e \in E} p_A(e|x) \log_2 \left(\frac{2p_A(e|x)}{p_A(e|x=1) + p_A(e|x=-1)} \right) de \\ 0 \leq I_A \leq 1 \quad (6.20a)$$

where p_A is the probability density function of L_a . For Gaussian noise, Eq. (6.20a) can be rewritten as

$$I_A = 1 - \int \exp \left(\frac{-1}{2\sigma_a^2} \left(y - \frac{\sigma_a^2}{2} \right)^2 \right) \frac{\log_2(1 + e^{-y})}{\sqrt{2\pi\sigma_a}} dy \quad (6.20b)$$

$$I_E = \frac{1}{2} \int \sum_{e \in E} p_E(e|x) \log_2 \left(\frac{2p_E(e|x)}{p_E(e|x=1) + p_E(e|x=-1)} \right) de \\ 0 \leq I_E \leq 1 \quad (6.21)$$

where p_E is the probability density function of L_e , viewing I_E as a function of I_E and $\frac{E_b}{N_o}$, the extrinsic information transfer characteristics are defined as

$$I_E = T \left(I_A, \frac{E_b}{N_o} \right) \quad (6.22)$$

For fixed $\frac{E_b}{N_o}$, the above transfer characteristic can be rewritten as

$$I_E = T(I_A) \quad (6.23)$$

Once I_{A1} and I_{E1} for decoder 1 and I_{A2} and I_{E2} for decoder 2 are obtained using Eqs. (6.20a), (6.20b) and (6.21), they are drawn on a single chart that is I_{A1} on the x-axis and I_{E1} on the y-axis for decoder 1, and for decoder 2, I_{E2} on the x-axis and I_{A2} on the y-axis, resulting in EXIT chart for the turbo decoder.

The steps involved in obtaining EXIT curve can be summarized as follows:

1. Specify the turbo code rate R and interested $\frac{E_b}{N_o}$ and determine AWGN $N_o/2$.
2. Specify μ_a of interest and determine $\sigma_a^2 = 2\mu_a$.
3. Run the turbo code simulator which yields encoded bits $y = [xp_1 p_2]$;
4. Find L_a using Eq. (6.19), and probability density function $p_A(e|x)$ by using a histogram of L_a . Then, find mutual information I_A using Eq. (6.20a).
5. Run the BCJR decoder using the model $r = y + \eta$ with $\eta = \mathcal{N}(0, \sigma^2) \sigma^2 = N_o/2$. In addition to the r , the decoder has $L_c = \frac{2r}{\sigma^2}$, and L_a given by Eq. (6.19).
6. Find L_e and then determine the probability density function $p_E(e|x)$ from the histogram of L_e and calculate mutual information I_E using Eq. (6.21).

7. If all values for μ_a of interest are exhausted, then stop and plot I_A versus I_E . Otherwise, go to step 2.

The EXIT charts at $\frac{E_b}{N_o} = 0.5$ dB and -0.2 dB are shown in Fig. 6.26 for rate 1/3 turbo encoder considered in [1].

From Fig. 6.26, it is observed that the curves cross for the EXIT chart at $\frac{E_b}{N_o} = -0.2$ dB and the turbo decoder does not converge. Hence, the ensemble threshold for rate 1/3 turbo encoder [1] must be at around -0.2 dB.

The EXIT charts at $\frac{E_b}{N_o} = -5$ dB and -4 dB are shown in Fig. 6.27 for rate 1/3 UMTS turbo encoder considered in Example 6.2.

Fig. 6.26 EXIT charts for the rate 1/3 turbo encoder [1] at $\frac{E_b}{N_o} = 0.5$ dB and -0.2 dB

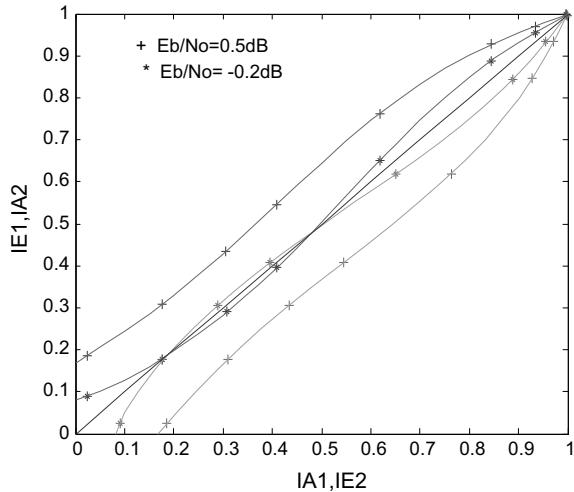
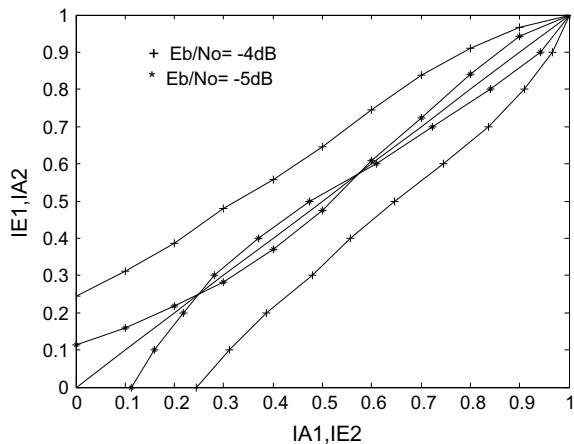


Fig. 6.27 EXIT charts for the rate 1/3 UMTS turbo encoder at at $\frac{E_b}{N_o} = -4$ dB and -5 dB



From Fig. 6.27, it is observed that the curves cross for the EXIT chart at $\frac{E_b}{N_0} = -5$ dB and the turbo decoder does not converge. Hence, the ensemble threshold for rate 1/3 UMTS turbo encoder must be at around -5 dB.

For a given code and channel, the decoding correctness of turbo decoder can check by examining whether the decoders EXIT curves cross. The ensemble threshold can be estimated by finding the $\frac{E_b}{N_0}$ for which the EXIT curves of the decoders cross. The speed of the decoder can be obtained from EXIT curves. The wider the gap between the EXIT curves of the two decoders, fewer the number of iterations required for convergence.

6.4 Performance Analysis of the Turbo Codes

6.4.1 Upper Bound for the Turbo Codes in AWGN Channel

Assuming that the transmitted data symbols are BPSK modulated which are coherently demodulated at the receiver, bit error probability bounds for turbo codes on AWGN channels can be upper bounded by the union bound [8, 9]

$$\text{BER}_{\text{ub}} \leq \sum_{w=1}^N \sum_{d=d_f}^{\infty} A(w, d) \frac{w}{N} Q\left(\sqrt{2 \cdot d \cdot R \cdot \frac{E_b}{N_0}}\right) \quad (6.24)$$

where $A(w, d)$ is the number of codeword of input weight w and the total weight d . The code's block size is given by the number of information bits N and the code rate R . Ignoring the effect of the tail (assuming that the tail length N), we can use our usual definition of N as being the length of the whole source sequence, including the tail. Thus, changing the order of summation:

$$\text{BER}_{\text{ub}} \leq \sum_{d=d_f}^{\infty} \left[\sum_{w=1}^N A(w, d) \frac{w}{N} \right] Q\left(\sqrt{2 \cdot d \cdot R \cdot \frac{E_b}{N_0}}\right) \leq \sum_{d=d_f}^{\infty} A_d \cdot Q\left(\sqrt{2 \cdot d \cdot R \cdot \frac{E_b}{N_0}}\right) \quad (6.25)$$

where A_d is the total information weight of all codewords of weight d divided by the number of information bits per codeword, as defined by

$$A_d = \sum_{w=1}^N A(w, d) \frac{w}{N} \quad (6.26)$$

Now, define N_d to be the number of codewords of the total weight d and w_d to be their average information weight. Thus,

$$N_d \cdot w_d = \sum_{w=1}^N A(w, d) \cdot w \quad (6.27)$$

$$A_d = w_d \frac{N_d}{N} \quad (6.28)$$

where $\frac{N_d}{N}$ is called the *effective multiplicity* of codewords of weight d . Substituting Eq. (6.28) in Eq. (6.25), we obtain

$$\text{BER}_{\text{ub}} \leq \sum_{d=d_f}^{\infty} w_d \frac{N_d}{N} Q\left(\sqrt{2 \cdot d \cdot R \cdot \frac{E_b}{N_0}}\right) \quad (6.29)$$

6.4.2 Upper Bound for Turbo Codes in Rayleigh Fading Channel

In MRC, the receiver weights the incoming signals on antennas by the respective conjugates of the complex fading random variables. The pairwise bit error probability with MRC in a Rayleigh fading channel for BPSK case is given by [10]

$$P_{d,MRC} = \frac{1}{\pi} \int_{\theta=0}^{\frac{\pi}{2}} \left[\frac{\sin^2 \theta}{\sin^2 \theta + \frac{E_b}{N_0}} \right]^{Ld} d\theta \quad (6.30)$$

Recalling the result in [11]

$$\frac{1}{\pi} \int_{\theta=0}^{\frac{\pi}{2}} \left[\frac{\sin^2 \theta}{\sin^2 \theta + c} \right]^n d\theta = \pi [P_e]^n \sum_{k=0}^{n-1} \binom{n-k+1}{k} (1-P_e)^k \quad (6.31)$$

where

$$P_e = \frac{1}{2} \left(1 - \sqrt{\frac{\frac{E_b}{N_0}}{1 + \frac{E_b}{N_0}}} \right)$$

Using Eq. (6.31) in Eq. (6.30), in closed form we obtain

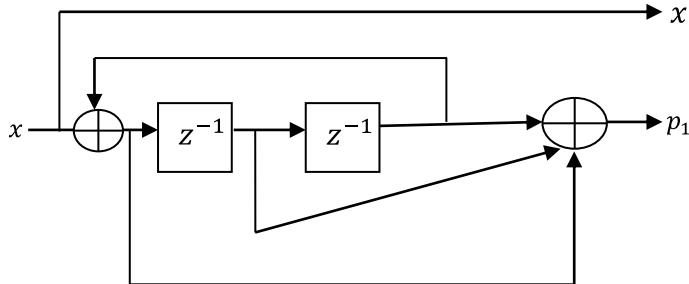


Fig. 6.28 RSC encoder of Example 6.7

$$P_{d,\text{MRC}} = [P_e]^{Ld} \sum_{k=0}^{Ld-1} \binom{Ld-1+k}{k} (1-P_e)^k \quad (6.32)$$

Then, the upper bound on the BER performance of turbo codes in Rayleigh fading channel with MRC diversity can be expressed as

$$\text{BER}_{\text{Rayleigh,MRC}} \leq \sum_{d=d_f}^{\infty} A_d P_{d,\text{MRC}} \quad (6.33)$$

If there is no diversity, i.e., $L = 1$, the upper bound on the BER performance of turbo codes in Rayleigh fading channel can be expressed as

$$\text{BER}_{\text{Rayleigh}} \leq \sum_{d=d_f}^{\infty} A_d P_d \quad (6.34)$$

where

$$P_d = [P_e]^d \sum_{k=0}^{d-1} \binom{d-1+k}{k} (1-P_e)^k$$

Example 6.7 Consider a turbo encoder using the following recursive systematic convolutional encoder shown in Fig. 6.28 with free distance 5, plot the upper bound BER versus $\frac{E_b}{N_o}$ performance of the turbo encoder for interleaver length of 100 in AWGN, Rayleigh fading channel with MRC diversity for $L = 2$.

Solution The set of coefficients A_d used to compute the bound for interleaver length of 100 as quoted in [8] are given as follows:

d	A_d	d	A_d
8	0.039881	22	33.31
9	0.079605	23	54.65
10	0.1136	24	91.23
11	0.1508	25	154.9
12	0.1986	26	265.5
13	0.2756	27	455.6
14	0.4079	28	779.0
15	0.6292	29	1327.0
16	1.197	30	2257.0
17	2.359	31	3842.0
18	4.383	32	6556.0
19	7.599	33	11221.0
20	12.58	34	19261.0
21	20.46	35	33143.0

The following MATLAB program is written and used to plot the $\frac{E_b}{N_o}$ versus upper bound BER performance of the turbo encoder with an interleaver length of 100 (Fig. 6.29).

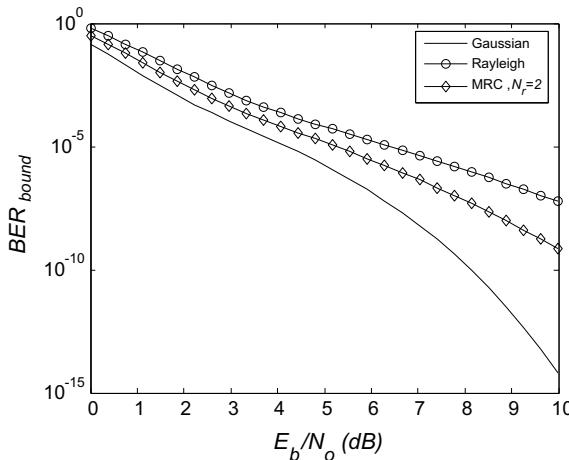


Fig. 6.29 Upper bound BER performance of turbo encoder of Example 6.7 with interleaver length of 100

Program 6.3: MATLAB program to compute upper bound BER for different $\frac{E_b}{N_o}$

```

clear all;close all
gamma_b_in_dB = [0:0.37:10];gamma_b = 10.^((gamma_b_in_dB/10));
d = [8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 ];
R_c =1/3;
Ad=[0.0039881 0.0079605 0.011918 0.015861 0.019887 0.024188
0.029048 0.034846 0.065768 0.1457 0.2984 0.5472 0.9171 1.437 2.144
3.09 4.465 6.716 10.67 17.65 29.61 49.31 80.57 128.6 201.3 311.5 481.2
748.8];
for i=1:28
sumr=0,summr=0;
q1=sqrt((gamma_b(i)*R_c)/(1+gamma_b(i)*R_c));
q2=sqrt((0.5*gamma_b(i)*R_c)/(1+0.5*gamma_b(i)*R_c));
for j=1:28
d1=d(j);q=0.5*(1-q1);q22=0.5*(1-q2);pd1=0;pd2=0;
for k=0:d1-1
pd1=pd1+nchoosek((d1-1+k),k)*((1-q)^k);
end
for k=0:2*d1-1
pd2=pd2+nchoosek((2*d1-1+k),k)*((1-q22)^k);
end
pd=(q^d1)*pd1;sumr=sumr+Ad(j)*pd;pdm=q22^(2*d1)*pd2;
summr=summr+Ad(j)*pdm;
sum=sum+ Ad (j)* 0.5* erfc ( sqrt( d (j)* R_c * gamma_b(i) ) );
end
BER_boundg(i)=sum; BER_boundray(i)=sumr;
BER_boundraym(i)=summr;
end
figure
semilogy(gamma_b_in_dB(1:28),BER_boundg, '-');
hold on
semilogy(gamma_b_in_dB(1:28),BER_boundray, '-o');
semilogy(gamma_b_in_dB(1:28),BER_boundraym, '-d');
xlabel('itE_b/N_o (dB)'), ylabel('itBER_b_o_u_n_d');
legend('Gaussian','Rayleigh','MRC ,itN_r=2');

```

6.4.3 Effect of Free Distance on the Performance of the Turbo Codes

The BER performance of turbo codes for ML decoding is upper bounded by Eq. (6.29). Since the BER performance of the code is dominated by the free distance term (for $d = d_f$) for moderate and high SNRs, for AWGN channel, Eq. (6.29) can be written as [12]

$$\text{BER}_{d_f, \text{AWGN}} \approx w_f \cdot \frac{N_f}{N} \cdot Q\left(\sqrt{2 \cdot d_f \cdot R \cdot \frac{E_b}{N_0}}\right) \quad (6.35)$$

where N_f and w_f correspond to N_d and w_d for $d = d_f$.

$$\text{BER}_{d_f, \text{Rayleigh}} \approx w_f \cdot \frac{N_f}{N} \cdot \sum_{k=0}^{d_f-1} \binom{d_f - 1 + k}{k} \left(1 - \frac{1}{2} \left(1 - \sqrt{\frac{\frac{E_b}{N_0} R}{1 + \frac{E_b}{N_0} R}}\right)\right)^k \quad (6.36)$$

For a turbo code considered in [13], the BER performance is evaluated in AWGN and Rayleigh fading channels with a pseudorandom interleave of length with 65536 with $N_f = 3$, $d_f = 6$, $w_f = 2$ and a 120×120 rectangular window with $N_f = 28900$, $d_f = 12$, $w_f = 4$.

The following MATLAB program is written and used to evaluate the performance in AWGN and Rayleigh fading channels.

Program 6.4 MATLAB program for free distance asymptotes in AWGN and Rayleigh fading channels for two different interleavers

```
clear all
close all
gamma_b_in_dB = [0:0.5:4];
gamma_b = 10.^((gamma_b_in_dB/10));
N_free = 3;
tilde_w_free = 2;
d_free = 6;
R_c = 1/2;
N = 65536;
N_freerec = 28900;
tilde_w_freerec = 4;
d_freerec = 12;
R_c = 1/2;
Nrec = 14400;
d1=d_free;
```

```

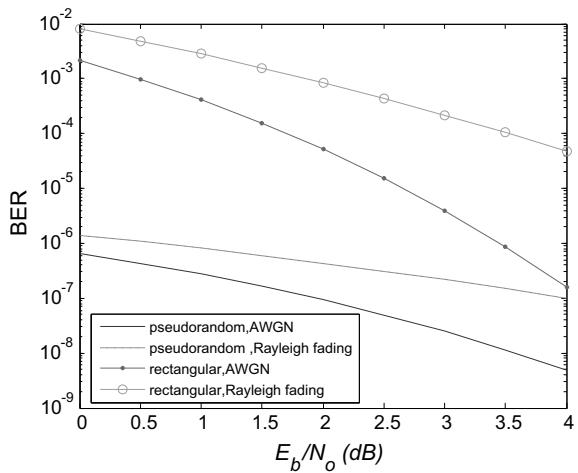
P_b_free_pseudorandom = N_free * tilde_w_free /N * 0.5* erfc ( sqrt( d_free
* R_c * gamma_b ) );
P_b_free_rectangular = N_freerec * tilde_w_freerec /Nrec * 0.5* erfc ( sqrt(
d_freerec * R_c * gamma_b ) );
for i=1:length(gamma_b_in_dB)
    q1=sqrt((gamma_b(i)*R_c)/(1+gamma_b(i)*R_c));
    q=0.5*(1-q1);
    pd1=0;
    for k=0:d1-1
        pd1=pd1+nchoosek((d1-1+k),k)*((1-q)^k);
    end
    P_b_free_pseudorandomray(i) =( N_free * tilde_w_free /N)*(q^(d1))*pd1;
end
d1=d_freerec;
for i=1:length(gamma_b_in_dB)
    q1=sqrt((gamma_b(i)*R_c)/(1+gamma_b(i)*R_c));
    q=0.5*(1-q1);
    pd1=0;
    for k=0:d1-1
        pd1=pd1+nchoosek((d1-1+k),k)*((1-q)^k);
    end
P_b_free_rectangulararray(i) =( N_freerec * tilde_w_freerec
/Nrec)*(q^(d1))*pd1;
end
figure
semilogy(gamma_b_in_dB, P_b_free_pseudorandom, '-' ,gamma_b_in_dB, P_b_free_rectangular, '-.
',gamma_b_in_dB, P_b_free_rectangulararray, '-o');
xlabel('itE_b/N_o (dB)'), ylabel('BER');
Legend('pseudorandom,AWGN','pseudorandom ,Rayleigh fading','rectangular,AWGN','rectangular,Rayleigh fading');

```

The free distance asymptotes for pseudorandom and rectangular interleavers in AWGN and Rayleigh fading channels obtained from the above MATLAB program are shown in Fig. 6.30.

From Fig. 6.30, it can be observed that the rectangular window exhibits relatively poor performance. It is due to the fact that the rectangular window has large effective multiplicity as compared to the effective multiplicity of the pseudorandom interleaver.

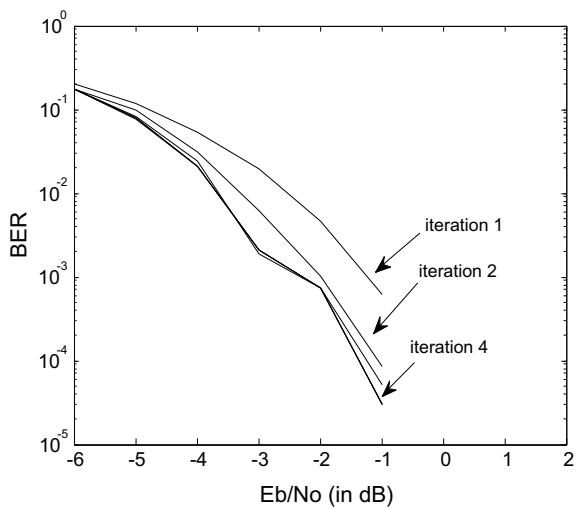
Fig. 6.30 Free distance asymptotes for turbo codes in AWGN and Rayleigh fading channels with two different interleavers



6.4.4 Effect of Number of Iterations on the Performance of the Turbo Codes

The BER performance of UMTS turbo codes for frame length of 40 is shown in Fig. 6.31. It can be seen that as the number of iterations increases, there is a significant improvement in BER performance. However, certain number iterations, no improvement can be observed. For complexity reasons, in turbo decoding, 4 to 10 iterations are used.

Fig. 6.31 Effect of number of iterations on the BER performance of turbo codes



6.4.5 Effect of Puncturing on the Performance of the Turbo Codes

The BER performance comparison of the unpunctured and the punctured turbo codes is shown in Fig. 6.32. For this, a turbo encoder is considered that uses RSC encoders with the generating function.

$$G = \begin{bmatrix} 1 & \frac{1+D^2}{1+D+D^2} \end{bmatrix}$$

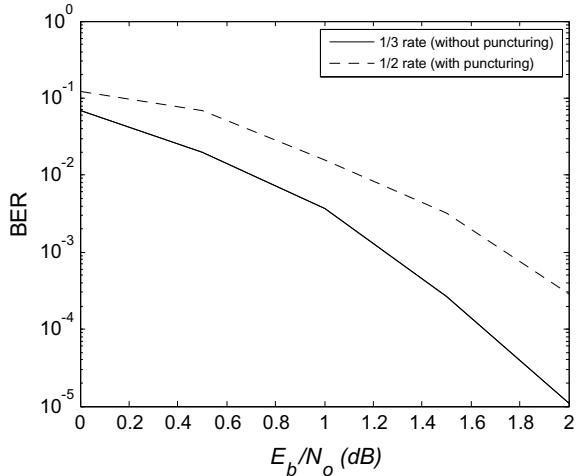
A random interleaver of length 1000 with odd–even separation is used. An AWGN channel with BPSK modulation assumed. In the decoding, Log BCJR algorithm for 3 iterations is used.

From Fig. 6.32, it can be observed that the unpunctured turbo codes give a gain of about 0.6 dB over the punctured turbo codes.

6.5 Enhanced Turbo Codes

Turbo codes will exhibit error floor down to block error rate (BLER) of 10^{-3} to 10^{-4} . The enhanced turbo codes [14] with list decoding do not exhibit error floor down to BLER of 10^{-3} to 10^{-4} .

Fig. 6.32 Effect of puncturing on the BER performance of turbo codes



6.5.1 Enhanced Turbo Encoder

The enhanced turbo encoder is a rate compatible conventional turbo encoder in which the interleaver parameters are kept constant for all the code rates and puncturing is varied according to the coding rate. It consists of two identical recursive systematic convolutional (RSC) encoders working with parallel concatenation as shown in Fig. 6.1. The RSC encoder 1 generates low-weight parity bits (p_1) from the input sequence x . The RSC encoder 2 produces high-weight parity bits (p_2) from input sequence scrambled by the interleaver. The combination of thus generated low-weight parity bits and high-weight parity bits generates a moderate weight turbo code. The original input sequence x and parity bits p_1 and p_2 are transmitted over the channel.

The interleaving function for enhanced turbo codes is given by

$$\pi(i) = (Pi + S(i \bmod Q)) \bmod K \quad (6.37)$$

where i stands for the data symbol address after interleaving. $\pi(i)$ denotes the corresponding address before interleaving

6.5.2 Enhanced List Turbo Decoder

The extrinsic information is set to zero in the first iteration using the conventional turbo decoder. Instead of using zero, if some values are used for the extrinsic information, different decoding results can be obtained. However, it may not yield correct decoding results. Hence, a list of decoded words can be obtained by using several enhanced turbo decoders in parallel [15] as shown in Fig. 6.33. The enhanced turbo decoders are initialized by their own a priori information and a list of decoded words will be obtained. After that correct one is chosen. In the list decoding technique for decoding enhanced turbo codes, a list Viterbi decoder [16] is applied to find a list containing the L most likely paths.

6.6 Performance Analysis of Enhanced Turbo Codes

6.6.1 Performance of Enhanced Tail-Biting Turbo Codes Over AWGN and Rayleigh Fading Channels

The BLER performance of the enhanced tail-biting turbo codes over AWGN and Rayleigh fading channels using QPSK modulation for information block of 400 bits at code rate $R = 1/3$ (with no puncturing) with 8 iterations is shown in Fig. 6.34

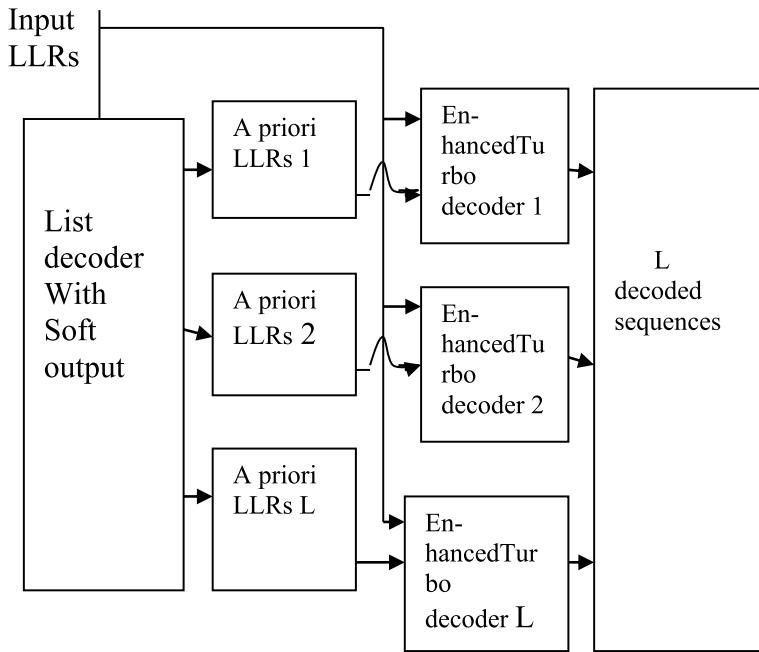
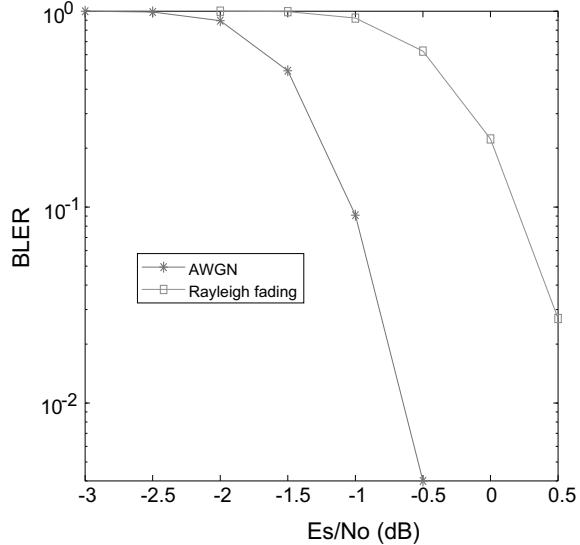


Fig. 6.33 Block diagram of enhanced list turbo decoding

Fig. 6.34 BLER performance of enhanced turbo codes with tail-biting decoding for $K = 400$ at $R = 1/3$



For this, a turbo encoder is considered that uses RSC encoders with the generating function.

$$\mathbf{G} = \left[\mathbf{1} \quad \frac{\mathbf{1} + \mathbf{D} + \mathbf{D}^3}{\mathbf{1} + \mathbf{D}^2 + \mathbf{D}^3} \right]$$

with the following interleaver parameters is used

ARP interleaver parameters ($K = 400$)

Q	P	$(S(0) \dots S(Q - 1))$
16	383	(8, 80, 311, 394, 58, 55, 250, 298, 56, 197, 280, 40, 229, 40, 136, 192)

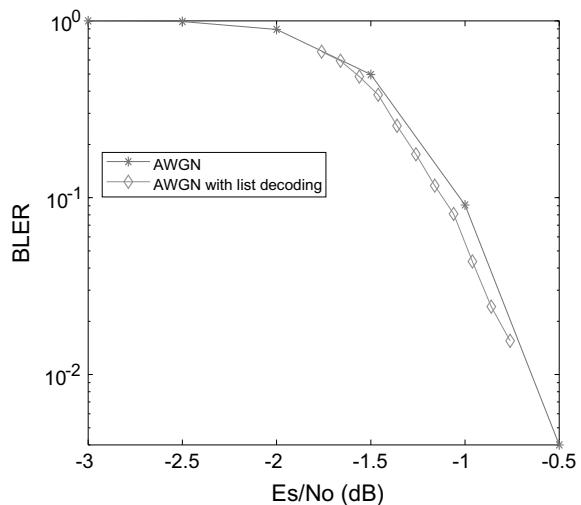
6.6.2 Performance of Enhanced Turbo Codes with Tail-Biting List Decoding in AWGN Channels

The enhanced turbo encoder considered in sub Sect. 6.6.1 is used and tail-biting turbo decoding is performed with list size 8.

The BLER performance of the enhanced tail-biting turbo codes over AWGN and Rayleigh fading channels using QPSK modulation for information block of 400 bits at code rate $R = 1/3$ with 8 iterations is shown in Fig. 6.35.

From Fig. 6.35, it is observed that tail-biting turbo list decoding performance is better than that of tail-biting turbo decoding.

Fig. 6.35 BLER performance of enhanced turbo codes with tail-biting decoding for $K = 400$ at $R = 1/3$ with list size = 8



6.7 Problems

1. For the encoder shown in Fig. 6.36
 - (i) Find the impulse response
 - (ii) Draw the state diagram
 - (iii) Obtain its equivalent recursive encoder
 - (iv) Find the impulse response of the recursive encoder obtained
 - (v) Draw the state diagram of the recursive encoder obtained.
2. Draw the equivalent RSC encoder of the convolutional encoder with generator sequences $g_1 = [1 \ 1 \ 1 \ 1 \ 1]$; $g_2 = [1 \ 0 \ 0 \ 0 \ 1]$.
3. For the turbo encoder shown in Fig., find the code word for the input sequence $x = \{1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0\}$. Let the interleaver be $\{5 \ 3 \ 4 \ 0 \ 6 \ 2 \ 1\}$
4. For the turbo encoder shown in Fig., find the code word for the input sequence $x = \{1 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0\}$. Let the interleaver be $\{7 \ 5 \ 1 \ 2 \ 4 \ 3 \ 6 \ 0\}$
5. Consider the CDMA2000 standard encoder shown in Fig. 6.37. Find the code word for the input sequence $x = \{1 \ 0 \ 1 \ 1 \ 0 \ 0\}$ assuming that the encoders trellis is terminated. Let the interleaver be $\{0 \ 3 \ 1 \ 5 \ 2 \ 4\}$.
6. Decode the following received sequence when the turbo code generated in Example 6.2 was transmitted over an AWGN channel with unity noise variance.

$x'(n)$	$p'_1(n)$	$p'_2(n)$
1.3209	2.4883	-1.3369
0.8367	-2.5583	-3.0404
-1.8629	-0.6555	-1.6638
-2.6356	0.3157	0.1266
-1.0138	0.9990	-1.3059
1.4864	0.6762	-1.0409
-2.2620	-0.7018	-0.5412

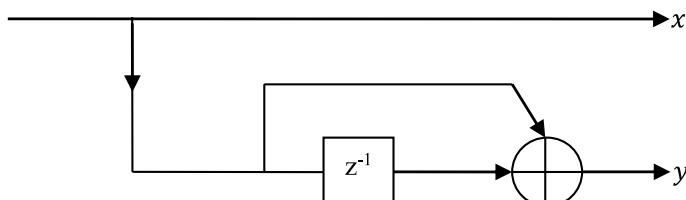


Fig. 6.36 Encoder of problem 1

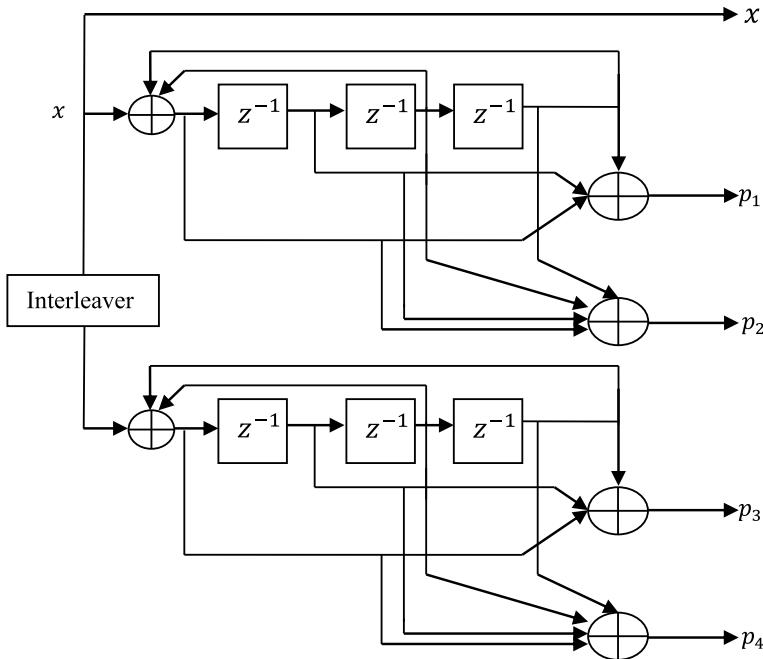


Fig. 6.37 CDMA2000 standard turbo encoder

6.8 MATLAB Exercises

1. Write a MATLAB program to construct an EXIT chart for turbo codes.
2. Write a MATLAB program to simulate the performance of unpunctured and punctured turbo codes.

References

1. C. Berrou, A. Glavieux, P. Thitimajshima, Near shannon limit error-correcting coding and decoding: turbo-codes, in *Proceedings of ICC 1993*, Geneva, Switzerland, pp. 1064–1070, May 1993
2. C. Berrou, A. Glavieux, Near optimum error correcting coding and decoding: turbo-codes. *IEEE Trans. Commun.* **44**(10), 1261–1271 (1996)
3. P. Jung, M. Nasshan, Performance evaluation of turbo codes for short frame transmission systems. *Electron. Lett.* **30**(2), 111–113 (1994)
4. L. Hanzo, T.H. Liew, B.L. Yeap, Turbo coding, turbo equalisation and space time coding for transmission over fading channels. *IEEE Press, John Wiley & Sons Ltd.*
5. L. Bahl, J. Cocke, F. Jelinek, J. Raviv, Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Trans. Inf. Theor.* **20** (1974)
6. W.E. Ryan, S. Lin, *Modern codes: classical and modern*, Cambridge University Press (2009)

7. S. ten Brink, Convergence behavior of iteratively decoded parallel concatenated codes. *IEEE Trans. Commun.* **49**(10), 1727–1737 (2001)
8. S. Benedetto, G. Montorsi, Unveiling turbo codes: some results on parallel concatenated coding schemes. *IEEE Trans. Info. Theory* **42**, 409–429 (1996)
9. D. Divsalar, S. Dolinar, R.J. McEliece, F. Pollara, Transfer function bounds on the performance of turbo codes. TDA Progress Report 42-122, JPL, Caltech, August 1995
10. A. Ramesh, A. Chockalingam, L.B. Milstein, Performance analysis of turbo codes on nakagami fading channels with diversity combining, WRL-IISc-TR-108, Wireless Research Lab Technical Report, Indian Institute of Science, Bangalore, January 2001
11. A. Goldsmith, M.S. Alouini, A unified approach for calculating error rates of linearly modulated signals over generalized fading channels. <http://systems.stanford.edu>
12. K.-L. Du, M.N.S. Swamy, Wireless communications: communication systems from RF subsystems to 4G enabling technologies, Cambridge University Press (2010)
13. C.B. Schlegel, L.C. Perez, *Trellis and Turbo Coding* (IEEE Press, Piscataway, NJ, 2004)
14. Orange and IMT, R1-1612938 Enhanced turbo codes for NR: performance evaluation for eMBB and URLLC, in 3GPP TSG RAN WG1 87, November 2016
15. A. Akmalikhodzhaev, A. Kozlov, New iterative turbo code list\decoders, in *IEEE Proceedings of International Symposium on Problems of Redundancy in Information and Control Systems* (2014)
16. N. Seshadri, C.E.W. Sundberg, List Viterbi decoding algorithms with applications. *IEEE Trans. Commun.* **42**(234), 313323 (1994)

Chapter 7

Bandwidth Efficient Coded Modulation



The block codes, convolutional, and turbo codes discussed in the previous chapters achieve performance improvement expanding the bandwidth of the transmitted signal. However, when coding is being applied to bandwidth-limited channels, coding gain is to be achieved without signal bandwidth expansion. The coding gain for bandwidth-limited channels can be achieved by a scheme called trellis-coded modulation (TCM). The TCM is a combined coding and modulation technique that increases the number of signals over the corresponding uncoded system to compensate for the redundancy introduced by the code for digital transmission over band-limited channels. The term “trellis” is due to the fact that the trellis diagram for the TCM schemes is similar to the trellis diagrams of binary convolutional codes. In TCM schemes, the trellis branches are labeled with redundant non-binary modulation signals rather than with binary code symbols. The TCM schemes employ multilevel amplitude and phase modulation, such as PAM, PSK, DPSK, or QAM in combination with a finite-state encoder which governs the selection of modulation signals to generate coded signal sequences. In the receiver, the received noisy signals are decoded using soft decision Viterbi or BCJR decoder.

In the TCM, the “free distance” (minimum Euclidean distance) between the coded modulation signals exceeds the minimum distance between the uncoded modulation signals, at the same information rate, bandwidth, and signal power.

The basic principle of the TCM and further descriptions of it were published in [1–5]: the TCM has seen rapid transition from the research to the practical use in 1984, when the international telegraph and telephone consultative committee (CCITT) has adopted the TCM scheme with a coding gain of 4 dB for use in the high-speed voice band modems for 9.6/12.4 Kbps standard [4, 6, 7].

The main idea in the TCM is to devise an effective method that performs mapping of the coded bits into the signal symbols so as to maximize the free distance between coded signal sequences. A method based on the principle of mapping by set partitioning was developed by Ungerboeck in [1]. This chapter describes the classical bandwidth efficient TCM, turbo TCM (TTCM), bit-interleaved coded modulation (BICM), and bit-interleaved coded modulation iterative decoding (BICM-ID) and compares their BER performance.

7.1 Set Partitioning

Set partitioning divides a signal set into the smaller sets with maximally increasing smallest intra-set distances. Finally, the obtained small signal constellations will be referred to as the “subsets.” Every constellation point is used only once, and if the subsets are used with equal probability, then the constellation points all appear with equal probability. The following two examples illustrate the set partitioning. The signal constellation is partitioned into the subsets that Euclidean minimum distance between signal symbols in a subset is increased with each partition.

Example 7.1 (Set partitioning of 4-PSK signal)

Euclidian distance in a signal constellation is the distance between different points in the constellation diagram with respect to reference point.

The 4-PSK signal constellation shown in Fig. 7.1 is partitioned as shown in Fig. 7.2. In the 4-PSK signal set, the signal symbols are located on a circle of radius 1 and having a minimum distance separation of $\Delta_0 = 2\sin(\frac{\pi}{4}) = \sqrt{2}$.

Finally, the last stage of the partition leads to four subsets and each subset contains a single signal symbol.

Example 7.2 (Set partitioning of 8-PSK signal)

The 8-PSK signal constellation shown in Fig. 7.3 is partitioned as shown in Fig. 7.4. In the 8-PSK signal set, the signal symbols are located on a circle of radius 1 and having a minimum distance separation of $\Delta_0 = 2\sin(\frac{\pi}{8}) = 0.765$. The eight symbols are subdivided into two subsets of four symbols each in the first partition with the minimum distance between two symbols increases to $\Delta_1 = 2\sin(\frac{\pi}{4}) = \sqrt{2}$.

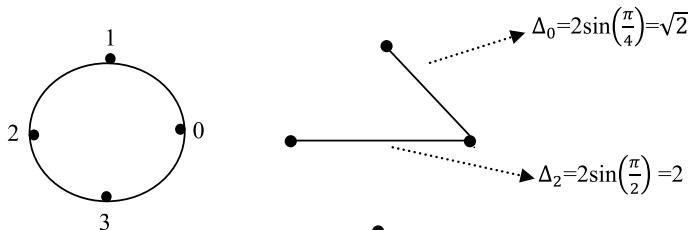


Fig. 7.1 Signal constellation diagram for the 4-PSK

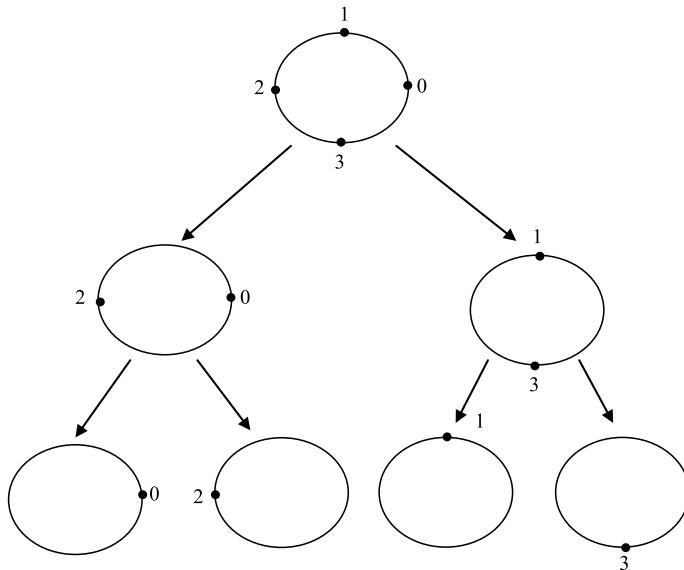


Fig. 7.2 4-PSK set partitioning

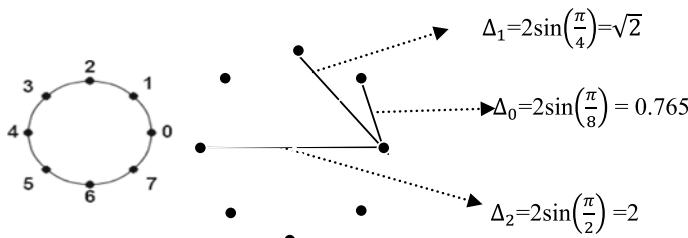


Fig. 7.3 Signal constellation diagram for the 8-PSK

Finally, the last stage of the partition leads to four subsets and each subset contains a single signal symbol.

7.2 Design of the TCM Scheme

The general structure of a TCM scheme is shown in Fig. 7.5. In an operation, k information bits are transmitted. The \tilde{k} ($\tilde{k} < k$) bits are encoded by binary convolutional encoder. The encoder output bits are used to select one of the possible subsets in the partitioned signal set, while the remaining $k - \tilde{k}$ bits are used to select one of $2^{m-\tilde{m}}$ signal symbols in each subset. When $\tilde{k} = k$, all the k information bits are encoded.

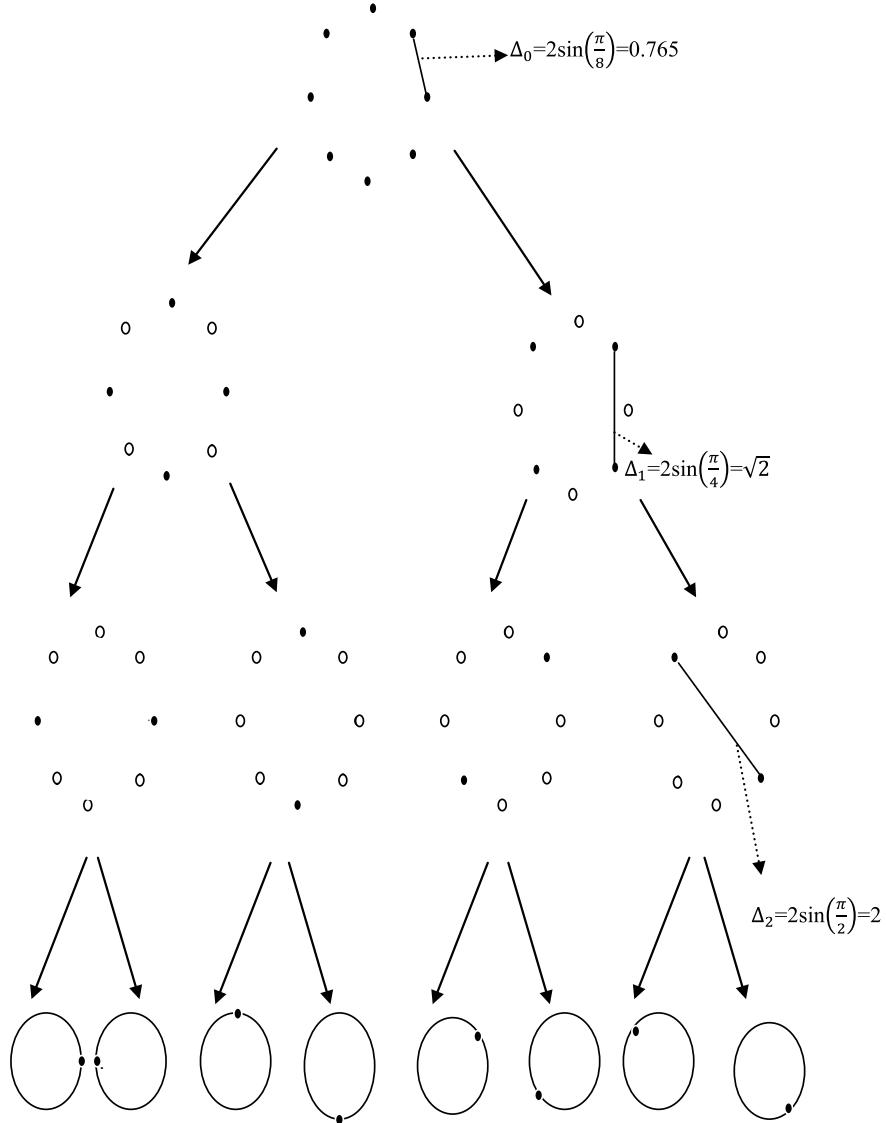


Fig. 7.4 8-PSK set partitioning

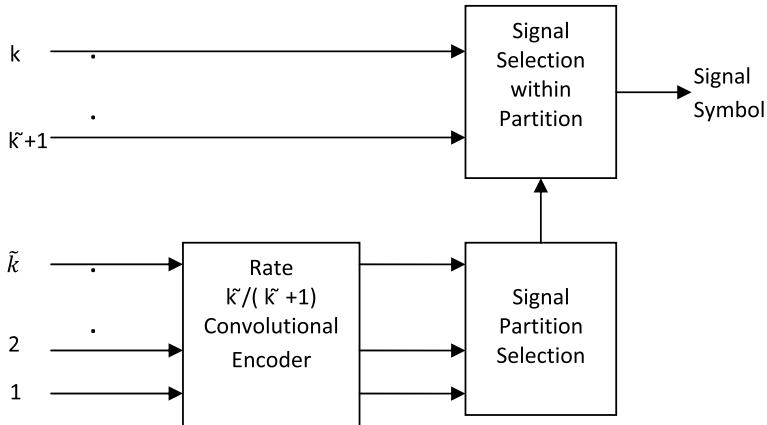


Fig. 7.5 General structure of a TCM scheme

In the encoder short designing, Ungerboeck summarized the following rules that were to be applied to the assigned channel signals

1. Transmission originating, or merging into any of the same state should receive signals from the subsets having maximum Euclidean distance between them.
2. Parallel state transitions are assigned the signal symbols separated by the largest Euclidean distance.
3. All the subsets are to be used with equal probability in trellis diagram.

The following examples illustrate the design of different TCM encoders

Example 7.3 (2-state 4-PSK TCM Encoder)

A simple 2-state 4-PSK TCM encoder is shown in Fig. 7.6a. In this encoder, a rate 1/2 convolutional encoder is used in which both the information bits are encoded. The output of the convolutional encoder is used to select from among the second-level partitions of 4-PSK, wherein each partition contains only a single signal. Thus,

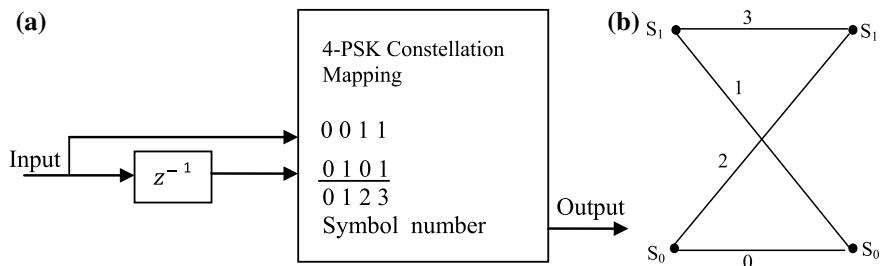
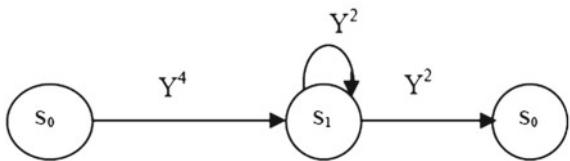


Fig. 7.6 a 2-state QPSK TCM encoder b 2-state QPSK TCM encoder trellis diagram

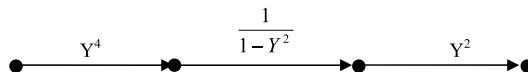
Fig. 7.7 Signal-flow graph of the trellis shown in Fig. 7.6b



it does not require an uncoded bit to complete the signal selection process. The two-state trellis diagram of the 4-PSK TCM encoder is shown in Fig. 7.6b, which has no parallel transitions.

The signal-flow graph of the trellis diagram of Fig. 7.6b is shown in Fig. 7.7. Now, the transfer function can be obtained by using the signal-flow graph techniques and Mason's formula. In the graph, the branch labels superscripts indicate the weight (SED) of the corresponding symbol of the transition branch in the trellis diagram.

By using reduction techniques, the above signal-flow graph can be simplified as follows:



Thus, the transfer function is given by

$$T(Y) = Y^4 \frac{1}{1 - Y^2} Y^2 = \frac{Y^6}{1 - Y^2}.$$

Example 7.4 (4-State 8-PSK TCM Encoder)

The Ungerboeck 4-state 8-PSK TCM encoder is shown in Fig. 7.8a. In this encoder, a rate $\frac{1}{2}$ convolutional encoder partitions the 8-PSK constellation into four subconstellations $\{(0, 4), (1, 5), (2, 6), (3, 7)\}$. The unique two-bit output from the convolutional encoder corresponds to a label assigned to each subconstellation. The output of the convolutional encoder selects one of the subconstellation, and the uncoded bit selects one of the two signals in the selected subconstellation.

The four-state trellis diagram of the TCM encoder is shown in Fig. 7.8b. In the trellis diagram, the states correspond to the contents of the memory elements in the convolutional encoder of the TCM encoder. The branch labels are the signals selected from the partitioned subconstellations for transmission associated with the given state transition. For example, if the convolutional encoder has to move from state S_0 to S_1 , then only signal 2 or 6 from subconstellation (2, 6) only may be selected for transmission.

The signal-flow graph of the trellis diagram of Fig. 7.8b is shown in Fig. 7.8c. The transfer function can be obtained by using the signal-flow graph techniques and Mason's formula.

The various distinct squared inter signal distances are as follows

$$\Delta_{0,1} = 2\sin\left(\frac{\pi}{8}\right) = 0.7654, \Delta^2(0, 1) = \Delta^2(000, 001) = 0.586$$

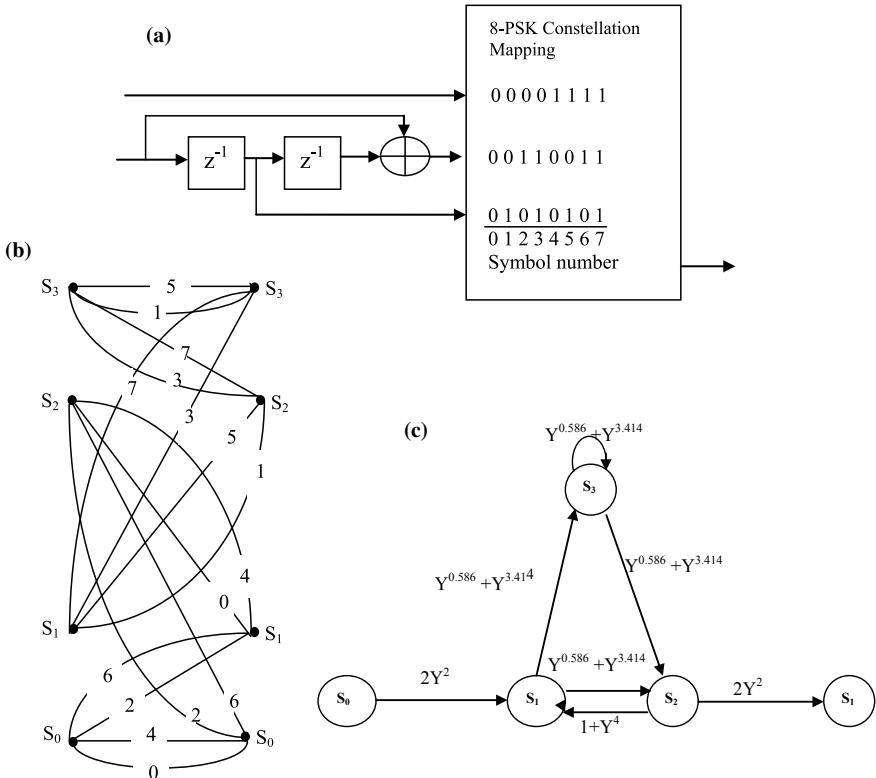


Fig. 7.8 **a** 4-state 8-PSK TCM encoder **b** 4-state 8-PSK TCM encoder Trellis **c** signal-flow graph of the trellis shown in Fig. 7.8b

$$\Delta_{0,2} = 2\sin\left(2 * \frac{\pi}{8}\right) = 1.4142, \quad \Delta^2(0, 2) = \Delta^2(000, 010) = 2.000$$

$$\Delta_{0,3} = 2\sin\left(3 * \frac{\pi}{8}\right) = 1.8478, \quad \Delta^2(0, 3) = \Delta^2(000, 011) = 3.414$$

$$\Delta_{0,4} = 2\sin\left(4 * \frac{\pi}{8}\right) = 2.0000, \quad \Delta^2(0, 4) = \Delta^2(000, 100) = 4.000.$$

By using the signal-flow graph reduction techniques and Mason's formula, we obtain the following transfer function.

$$T(Y) = 4 \frac{(Y^{4.586} + Y^{7.414})}{1 - 2Y^{0.586} - 2Y^{3.414} - Y^{4.586} - Y^{7.414}}$$

Example 7.5 (8-State 8-PSK TCM Encoder)

The Ungerboeck 8-state 8-PSK TCM encoder is shown in Fig. 7.9a. In this encoder, a rate 2/3 convolutional encoders used in which the both information bits

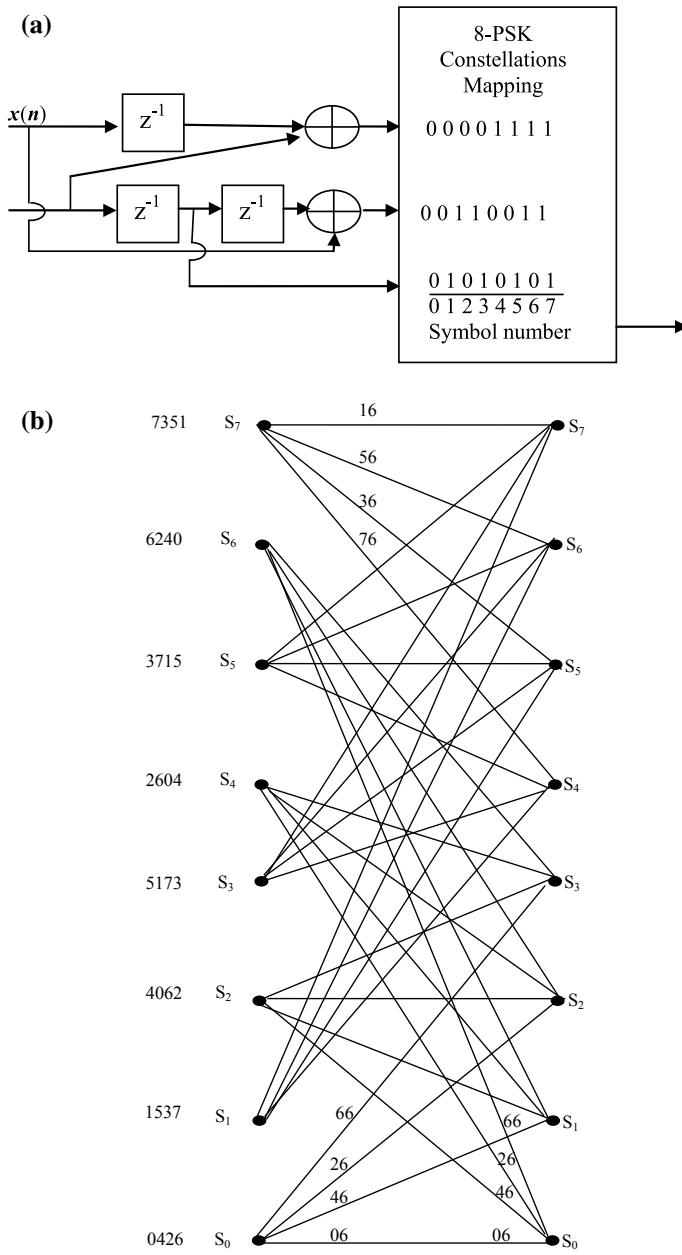


Fig. 7.9 **a** 8-state 8-PSK TCM encoder **b** 8-state 8-PSK TCM encoder trellis

are encoded. The output of the convolutional encoder is used to select from among the third-level partitions of 8-PSK, wherein each partition contains only a single signal. Thus, it does not require an uncoded bit to complete the signal selection process.

The 8-state trellis diagram of the 8-PSK TCM encoder is shown in Fig. 7.9b, which has no parallel transitions.

7.3 Decoding TCM

In general, a subconstellation of the signals is assigned to each branch in the TCM trellis. The decoding of the TCM is performed using the soft decision Viterbi algorithm in two steps.

1. Determine the best signal point within each subset by comparing the received signal to each of the signals allowed for a branch. The signal closest in distance to the received signal is considered as the best signal point and the corresponding branch metric is proportional to the distance between the best signal subset signal point and the received signal.
2. The signal point is selected from each subset and its squared distance is the signal path through the code trellis that has the minimum sum of squared distances from the received sequence.

7.4 TCM Performance Analysis

The performance of a TCM scheme can be evaluated by the following performance measures.

7.4.1 Asymptotic Coding Gain

The coded system performance improvement relative to the uncoded system is measured in terms of asymptotic coding gain. The asymptotic coding gain is defined as follows:

$$\text{Asymtotic coding gain} = \left(\frac{E_{\text{uncoded}}}{E_{\text{coded}}} \right) \left(\frac{d_{f/\text{coded}}^2}{d_{f/\text{uncoded}}^2} \right) \quad (7.1)$$

where E_{uncoded} is the normalized average received energy of an uncoded system, E_{coded} is the normalized average received energy of the coded system, $d_{f/\text{uncoded}}^2$ is

the squared minimum free distance of an uncoded system, and $d_{f/\text{coded}}^2$ is the squared minimum free distance of the coded system.

7.4.2 Bit Error Rate

A general lower bound for BER in an AWGN channel is

$$\Psi Q\left(\sqrt{\frac{d_f^2 E_s}{2N_0}}\right). \quad (7.2)$$

The distance structure is independent of the transmitted sequence for the uniform TCM and

$$\Psi = 1.$$

A closed form upper bound on BER can be expressed by

$$\text{BER}_{\text{UB}} = T(Y)|_{Y=\exp(-E_s/4N_0)} \quad (7.3)$$

or in a tighter form as

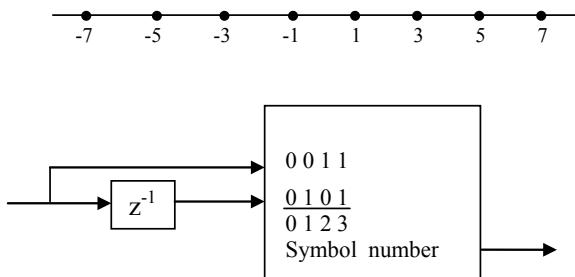
$$\text{BER}_{\text{UB}} = Q\left(\sqrt{\frac{d_f^2 E_s}{2N_0}}\right) \cdot \exp\left(\frac{d_f^2 E_s}{4N_0}\right) T(Y)|_{Y=\exp(-E_s/4N_0)} \quad (7.4)$$

where $T(Y)$ is the transfer function.

The following example illustrates the performance of a TCM scheme.

Example 7.6 Consider the following 2-state encoder and the 8-AM constellation to construct a TCM scheme that provides 2 bits/s/Hz. Determine the asymptotic coding gain for the TCM relative to the uncoded 4-AM system (Fig. 7.10).

Fig. 7.10 2-state 8-AM TCM encoder



Solution

Label 0 1 2 3 4 5 6 7.

For a 2^m -ary AM constellation with the same minimum free distance as BPSK, the normalized average energy is given by average signal energy

$$E = \frac{(4^m - 1)}{m} = \frac{(4^3 - 1)}{3} = \frac{63}{3} = 21$$

or

$$= \frac{1}{8}(1^2 + 3^2 + 5^2 + 7^2 + (-1)^2 + (-3)^2 + (-5)^2 + (-7)^2) = 21$$

8-AM Set Partitioning

See Fig. 7.11.

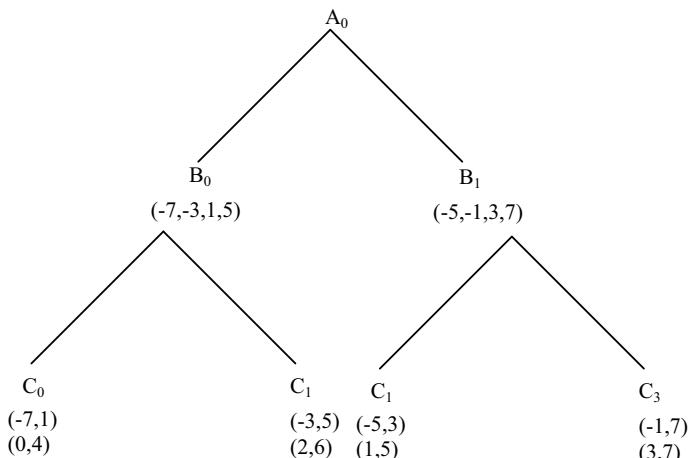
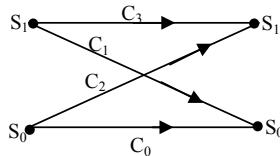
Trellis Diagram

Fig. 7.11 8-AM set partitioning

$$d_{f/\text{uncoded}} = 2$$

$$d_{f/\text{uncoded}} = \sqrt{\Delta_0^2 + \Delta_1^2} = \sqrt{2^2 + 4^2} = \sqrt{20}$$

since N-AM signal sets results in

$$E_{\text{uncoded}} = \frac{(4^m - 1)}{3} = \frac{(4^2 - 1)}{3} = 5$$

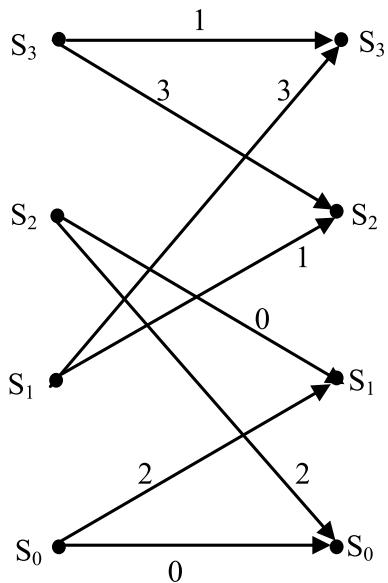
$$\text{coding gain} = \left(\frac{E_{\text{uncoded}}}{E_{\text{coded}}} \right) \left(\frac{d_{f/\text{coded}}^2}{d_{f/\text{uncoded}}^2} \right) = \frac{5}{21} \frac{20}{4} = 1.19 = 0.76 \text{ dB}$$

Example 7.7 Evaluate coding gain and the BER performance of a 4-state 4-PSK TCM with the following Trellis diagram.

The signal-flow graph of the trellis diagram of Fig. 7.12 is shown in Fig. 7.13. Now, by using the signal-flow graph reduction techniques and Mason's formula, the transfer function can be obtained.

By using reduction techniques, the above signal-flow graph can be simplified as

Fig. 7.12 Trellis diagram



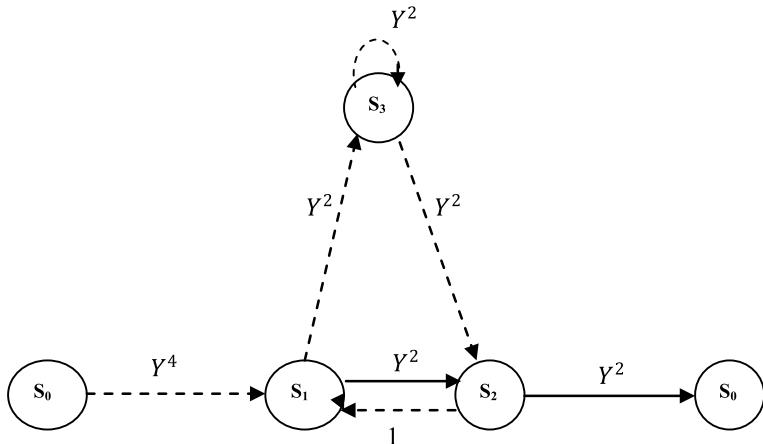
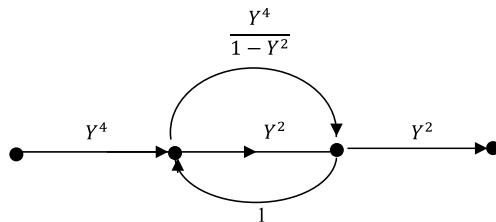
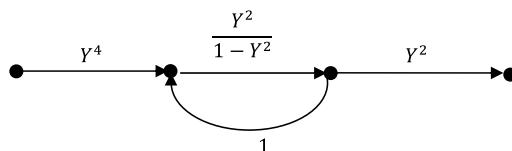


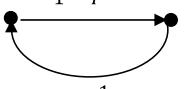
Fig. 7.13 Signal-flow graph



Further, the parallel branches with gains Y^2 and $\frac{Y^4}{1-Y^2}$ can be combined as a single branch with gain

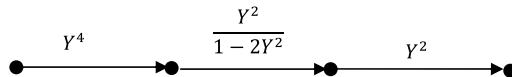
$$Y^2 + \frac{Y^4}{1-Y^2} = \frac{Y^2 - Y^4 + Y^4}{1-Y^2} = \frac{Y^2}{1-Y^2}$$



Further, the loop  can be replaced by a branch with gain

$$\frac{\frac{Y^2}{1-Y^2}}{1 - \frac{Y^2}{1-Y^2}} = \frac{Y^2}{1 - 2Y^2}$$

as follows



Thus, the transfer function is given by

$$T(Y) = Y^4 \frac{Y^2}{1 - 2Y^2} Y^4 = \frac{Y^{10}}{1 - 2Y^2}$$

Computation of d_f

See Fig. 7.14.

Since there are n_p parallel transitions in this trellis, only non-parallel paths are to be examined to determine minimum free distance of the code. At state S_0 , the symbol path 2 is chosen with the SED of 4, from there it leads us to state 1. From state S_1 , the symbol path 1 with the SED 2 is taken which takes to state S_2 . From state S_2 , we return to state S_0 via the symbol path 2 with SED of 4. There is no other path that can take us back to state S_0 with a smaller total SED.

Hence,

$$\begin{aligned} d_{f/\text{uncoded}}^2 &= \text{sum of the SEDs of the paths shown in bold} \\ &= 4 + 2 + 4 = 10. \end{aligned}$$

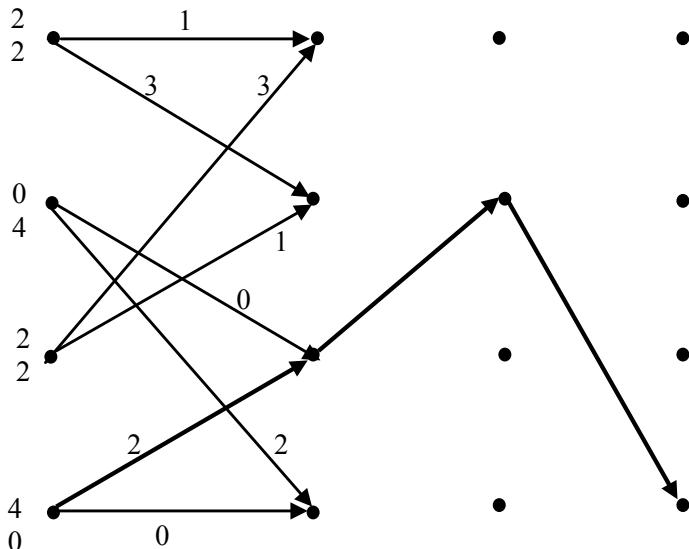
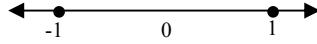


Fig. 7.14 Computation of d_f

Asymptotic coding gain of the 4-state 4-PSK TCM



Since BPSK constellation is with antipodal signals +1 and -1 as shown in above Figure. Thus,

$$d_{f/\text{uncoded}}^2 = 4$$

Hence, the asymptotic coding gain is given by

$$\text{Asymtotic coding gain} = \left(\frac{E_{\text{uncoded}}}{E_{\text{coded}}} \right) \left(\frac{d_{f/\text{coded}}^2}{d_{f/\text{uncoded}}^2} \right) = \frac{10}{4} = 2.5$$

BER Performance of the 4-state 4-PSK TCM

The transfer function bounded for the BER for the 4-state 4-PSK TCM in an AWGN channel from Eq. (7.3) is given by

$$\begin{aligned} \text{BER} &= T(Y)|_{Y=\exp(-E_b/4N_0)} = \frac{Y^{10}}{1-2Y^2} \Big|_{Y=\exp(-E_b/4N_0)} \\ &= \frac{\exp(-10E_b/4N_0)}{1-2\exp(-2E_b/4N_0)} \end{aligned}$$

The distance structure is independent of transmitted sequence for the uniform TCM and

$$\psi = 1$$

Since $d_f^2 = 10$ for the 4-state 4-PSK TCM, the lower bound for BER from Eq. (7.2) can be written as

$$\text{BER}_{\text{LB}} = Q\left(\sqrt{\frac{5E_b}{N_0}}\right)$$

The following MATLAB program illustrates the BER performance of 4-state 4-PSK TCM in comparison with uncoded BPSK (Fig. 7.15).

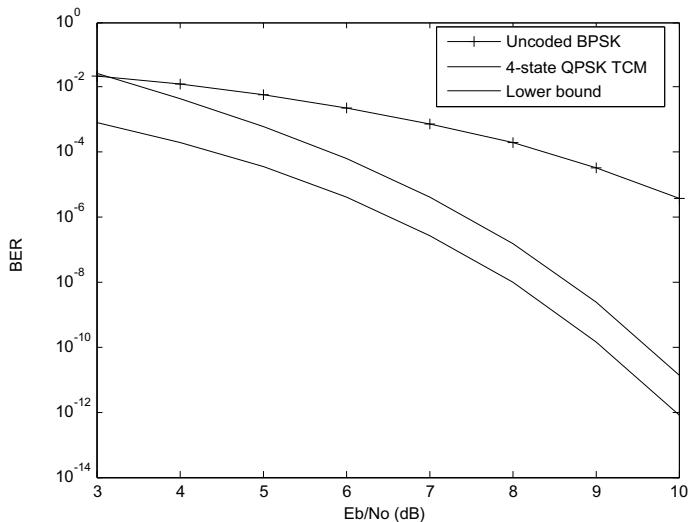


Fig. 7.15 BER performance comparison

Program 7.1: MATLAB program for BER performance of 4-state 4-PSK TCM

```
clear all;clc;
Eb_N0_dB=[3:1:10];
EbN0Lin = 10.^((Eb_N0_dB/10));
BER_BPSK_AWGN = 0.5* erfc ( sqrt( EbN0Lin ) );
BER_QPSK_LB = 0.5* erfc ( sqrt(2.5* EbN0Lin ) );
BER_QPSK =exp(-2.5*EbN0Lin)./(1-2*exp(-0.5*EbN0Lin));
semilogy(Eb_N0_dB,BER_BPSK_AWGN,'-+')
hold on
semilogy(Eb_N0_dB,BER_QPSK ,'-')
semilogy(Eb_N0_dB,BER_QPSK_LB,'--')
legend('Uncoded BPSK ','4-state QPSK TCM ','Lower bound');
xlabel('Eb/No (dB)');
ylabel('BER');
```

Example 7.8 Evaluate coding gain of a 4-state 8-PSK TCM scheme of Example 7.7

Solution

Computation of d_f

In the Trellis diagram shown in Fig. 7.16, symbols originating from a state are replaced with their SEDs.

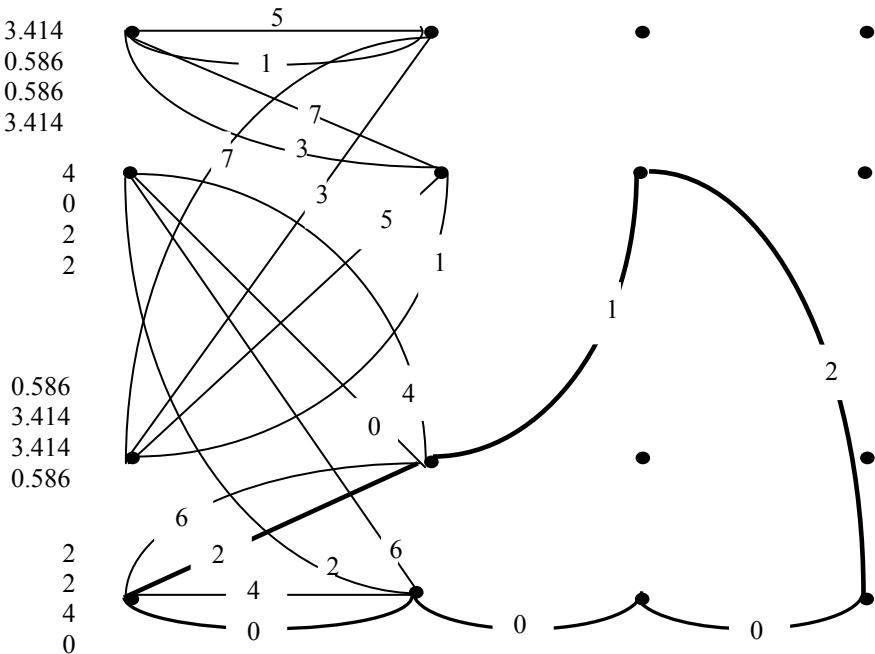


Fig. 7.16 Trellis diagram

Since there are n_p parallel transitions in this trellis, both the parallel and the non-parallel transitions are to be examined to determine minimum free distance of the code. The minimum free distance for the parallel transitions is the minimum free distance for the signals of the partition in the parallel transitions. For this encoder, the minimum free distance for the parallel transitions is the minimum free distance among $\{(0, 4), (1, 5), (2, 6), (3, 7)\}$. Hence,

$$d_{f/\text{parallel}} = 2$$

To compute $d_{f/\text{parallel}}$, the minimum distance path is found by following from each state the path with the smallest squared distance but not 0. At state S_0 , the symbol path 2 is chosen as it has the SED of 2, from there it leads us to state 1. From state 1, the symbol path 1 with the SED of 0.586 is taken which takes to state S_0 via the symbol path 2 with SED of 2.

There is no other path that can take us back to state S_0 with a smaller total SED. Thus, the total minimum squared Euclidean distance (MSED) is $2 + 0.586 + 2 = 4.586$ and hence the $d_{f/\text{nonparallel}} = \sqrt{4.586} = 2.14$. The minimum free distance for the TCM encoder is the minimum of $d_{f/\text{parallel}}$ and, $d_{f/\text{nonparallel}}$. Thus, the

$$d_{f/\text{coded}} = \min(d_{f/\text{parallel}}, d_{f/\text{nonparallel}}) = \min(2, 2.14) = 2.$$

The minimum free distance for the uncoded 4-PSK is $\sqrt{2}$ and so the $d_f^2 = 2$ for the uncoded 4-PSK. Therefore, the asymptotic coding gain for the 4-state 8-PSK TCM is given by

$$\text{coding gain} = 10 \log_{10} \left(\frac{d_{f/\text{coded}}^2}{d_{f/\text{uncoded}}^2} \right) = 10 \log_{10} \frac{4}{2} = 3.01 \text{ dB}.$$

Example 7.9 Evaluate coding gain of the 8-state 8-PSK TCM scheme of Example 7.5

Solution

Computation of d_f

In the trellis diagram shown in Fig. 7.17, symbols originating from a state are replaced with their SEDs. Ungerboeck encoder of Example 7.8, we have to compute d_f of this code in order to determine the asymptotic coding gain. The minimum distance path is found by following from each state the path with the smallest squared distance but not 0. At state S_0 , the symbol path 6 is chosen as it has the SED of 2, from there it leads us to state S_3 . From state S_3 , the symbol path 7 with the SED of 0.586 is taken which takes to state S_6 . From state S_6 , we return to state S_0 via the symbol path 6 with SED of 2.

There is no other path that can take us back to state S_0 with a smaller total SED. Thus, the total minimum squared Euclidean distance (MSED) is $2 + 0.586 + 2 = 4.586$ and hence the $d_f^2 = 4.586$ for the coded system. The minimum free distance for uncoded 4-PSK is $\sqrt{2}$ and so the $d_{\text{free}}^2 = 2$ for the uncoded 4-PSK. Therefore, the asymptotic coding gain for the 8-state 8-PSK TCM is given by

$$\text{coding gain} = 10 \log_{10} \left(\frac{d_{f/\text{coded}}^2}{d_{f/\text{uncoded}}^2} \right) = 10 \log_{10} \frac{4.586}{2} = 3.6 \text{ dB}$$

7.4.3 Simulation of the BER Performance of a 8-State 8-PSK TCM in the AWGN and Rayleigh Fading Channels Using MATLAB

The following MATLAB program7.2 and MATLAB functions given in Appendix A are used to simulate the BER performance of Ungerboeck 8-state 8-PSK TCM of Fig. 7.18 in both the AWGN and Rayleigh fading channels.

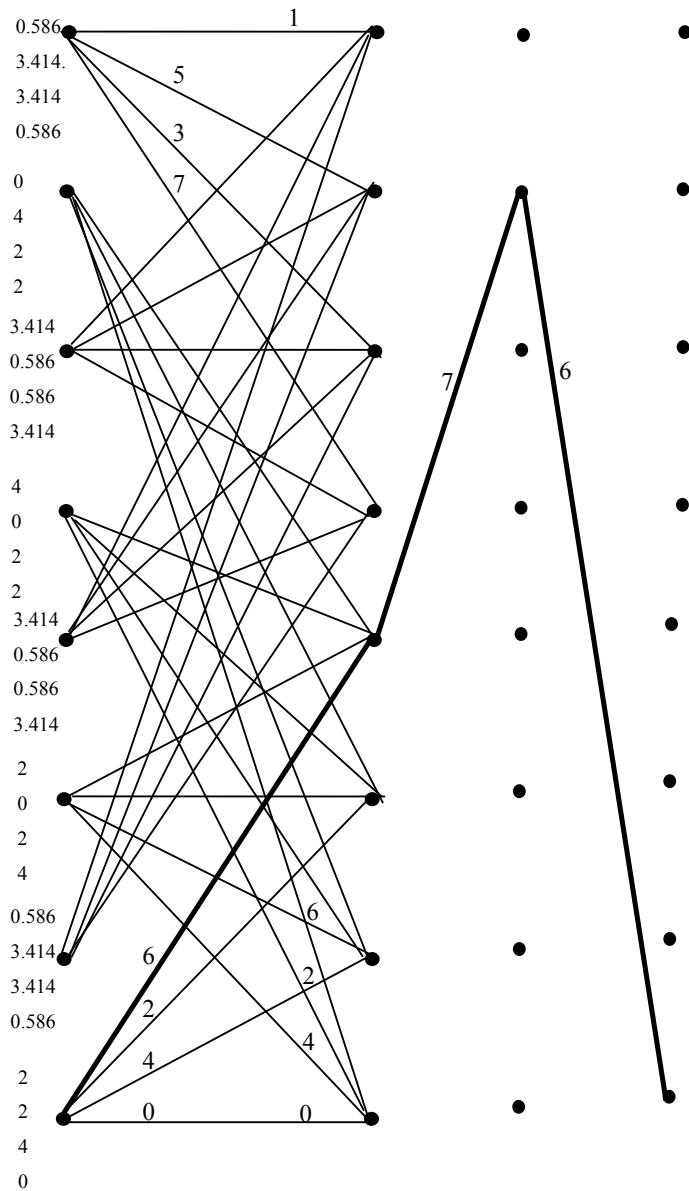


Fig. 7.17 Trellis diagram for Ungerboeck encoder shown in Fig. 7.16 replacing symbols with their SEDs

Program7.2

```
%MATLAB program to simulate BER performance of 8-state 8_PSK TCM in
AWGN%and Rayleigh fading channels
clear all; clc; close all;
global n k L nis M N S smap nl bps; global Cw Nes Prs
n=2;k=3;L=3;nis=512;
[Cw, Nes, Prs]=genpoly(n,k,L);%Generation of Trellis%
%Cw=codeword,Prs=previous state,Nes=next state
M=bitshift(1,n); N=nis; S=bitshift(1,L); [smap,bps,nl] = PSKmodSP(k);
Ec=1; EbN0dB=5; i=1; EbN0dB_stop=10;
nib=nis*n;% number of information bits
ncb=nis*k;% number of coded bits
while (EbN0dB <= EbN0dB_stop)
    errorsa=0; bitsa=0; errorsr=0;
    bitsr=0; frame=0;EbN0=10^(EbN0dB/10);
    while (errorsa < 1000 && frame<=100)
        Eb=Ec/((nib/ncb)*bps); %Eb =energy per bit
        N0=Eb*(EbN0^-1);%N0=variance
        inb=round(rand(1,nib));%inb=input bits
        symbols =bits2symbol(n,nis,inb);
        [Os,Ts]=tmemenc(symbols,Cw,Nes,smap);%Os=output sym-
        bols,Ts=Transmitted signal
        Rsa=Ts+ sqrt(N0/2)*(randn(size(Ts))+1i*randn(size(Ts))); %Rs=received
        signal in AWGN channel
        Rsr=Ts+sqrt(1/2)*0.3635*(randn(size(Ts))+1i*randn(size(Ts)))+
        sqrt(N0/2)*(randn(size(Ts))+1i*randn(size(Ts))); %Rs=received signal Ray-
        leigh fading channel
        Pra=demodsymbols(Rsa,N0);Prr=demodsymbols(Rsr,N0);
        decbitsa=bitsdecode(Pra);%decoded bits for AWGN
        decbitsr=bitsdecode(Prr);%decoded bits for Rayleigh
        errora=sum(decbitsa ~= inb);errorsa=errorsa+errora;
        errorr=sum(decbitsr ~= inb);errorsr=errorsr+errorr;
        bitsa=bitsa+sum(decbitsa ~= inb)+sum(decbitsa == inb);
        bitsr=bitsr+sum(decbitsr ~= inb)+sum(decbitsr == inb);frame=frame+1;
    end
    EbN0dB=EbN0dB+1;
    berawgn(i)=errorsa/bitsa; berray(i)=errorsr/bitsr; i=i+1; end
figure,
EbN0dB=[ 5 6 7 8 9 10];
berqpskawgn=BERAWGN(EbN0dB, 'psk', 4,'nondiff' );
berqpskfad=BERFADING(EbN0dB, 'psk', 4,1 );
semilogy(EbN0dB,berqpskfad ,'-*')
hold on
semilogy(EbN0dB,berqpskawgn,'-+')
semilogy(EbN0dB,berray,'-v')
semilogy(EbN0dB,berawgn,'d')
legend('Uncoded QPSK Rayleigh ','Uncoded QPSK AWGN ','8-state 8-PSK
Rayleigh','8-state 8-PSK AWGN'); xlabel('Eb/No in dB');ylabel('BER');
```

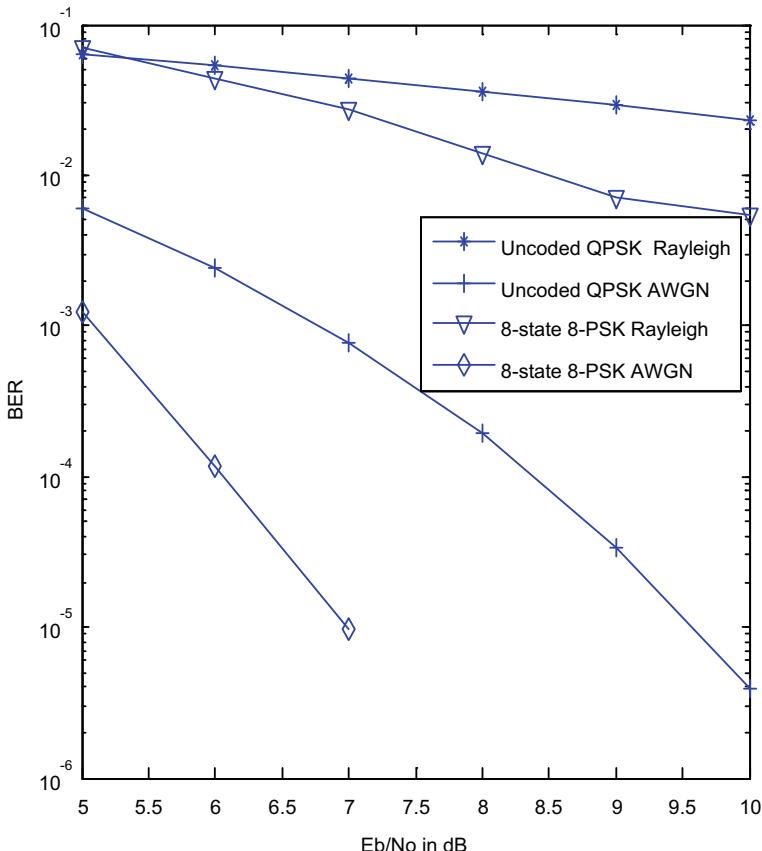


Fig. 7.18 BER performance

The BER performance obtained by using the programs for frame length of 512 bits for both the AWGN and the Rayleigh fading channels is shown in Fig. 7.18. The performance of the TCM in the AWGN channel is much better than the performance in the Rayleigh fading channel. The uncoded QPSK BER performance is also shown in Fig. 7.18 for both AWGN and Rayleigh fading channels, which will serve as reference to compare the performance of the coded modulation scheme in terms of coding gain.

7.5 Turbo Trellis-Coded Modulation (TTCM)

Robertson has introduced the concept of the “Turbo Trellis Coded Modulation (TTCM)” in [8] by using two recursive TCM encoders in parallel concatenation. The system overview for TTCM is shown in Fig. 7.19.

7.5.1 TTCM Encoder

TTCM encoder contains the parallel concatenation of two TCM encoders as shown in Fig. 7.20. Let the size of the interleaver be N . The number of modulated symbols per block is $N - n$, with $n = D/2$, where D is the signal set dimensionality. The number of information bits transmitted per block is $N - m$. The encoder is clocked in steps of $n \cdot T$. Where T is the symbol duration of each transmitted $2^{((m+1)/n)}$ -ary

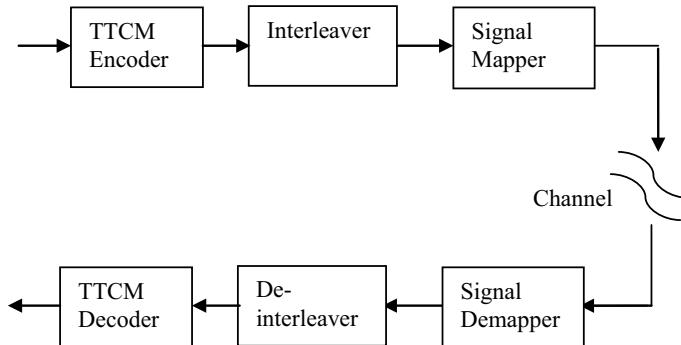


Fig. 7.19 System overview for TTCM

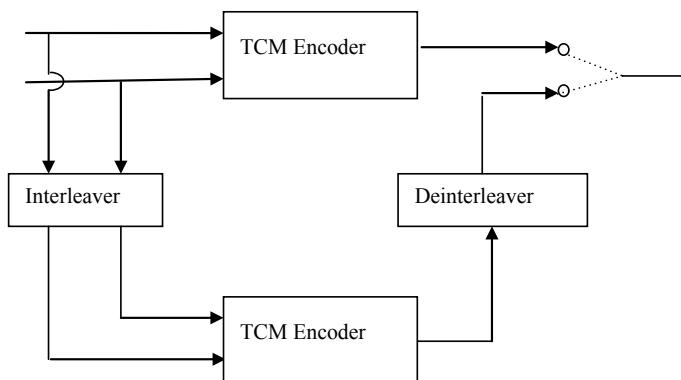


Fig. 7.20 TTCM encoder structure

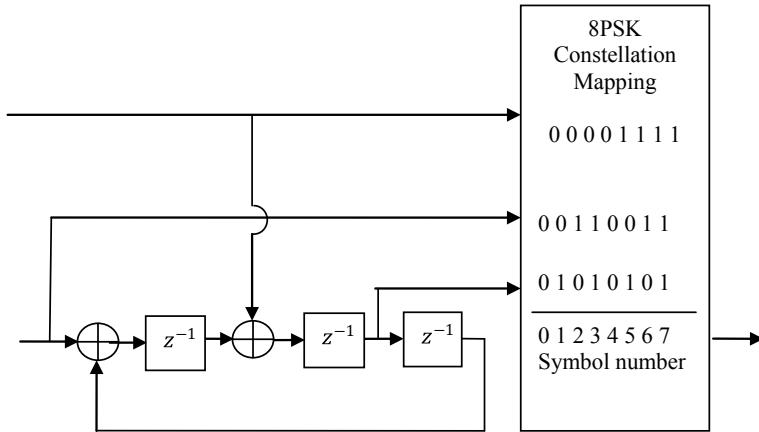


Fig. 7.21 TCM encoder used in TTCM encoder structure

symbol. In each step, m information bits are input and n symbols are transmitted, yielding a spectral efficiency of m/n bits per symbol usage. The first TCM encoder normally operates with the original bit sequence while the second encoder works with the interleaved version of the input bit sequence.

A simple example will now serve to clarify the operation of the TTCM encoder for the case of the following 8-state 8-PSK TCM with code rate 2/3 used in the TTCM encoder structure depicted in Fig. 7.21. A sequence of length 6 information bit pairs (00,01,11,10,00,11) is encoded by the first encoder to yield the 8-PSK sequence (0, 2, 7, 5, 1, 6). The information bits are interleaved on a pairwise basis using a random interleaver (3, 6, 5, 2, 1, 4) and encoded again into the sequence (6, 7, 0, 3, 0, 4) by the second encoder. We de-interleave the second encoder's output symbols to ensure that the ordering of the two information bits partly defining each symbol corresponds to that of the first encoder, i.e., we now have the sequence (0, 3, 6, 4, 0, 7). Finally, we transmit the first symbol of the first encoder, the second symbol of the second encoder, the third of the first encoder, the fourth symbol of the second encoder, etc., (0, 3, 7, 4, 1, 7). Thus, the transmitted signal will be of the symbols (0, 3, 7, 4, 1, 7).

7.5.2 TTCM Decoder

A block diagram of turbo decoder is shown in Fig. 7.22. The TTCM decoder is much similar to that of binary turbo codes, except the difference in the nature of the information passed from one decoder to other decoder, respectively, and the treatment of the very first decoding step. In symbol-based non-binary TTCM scheme, the systematic bit as well as the parity bits are transmitted together as in the form of complex enveloped symbol and cannot be separated from the extrinsic components, since the

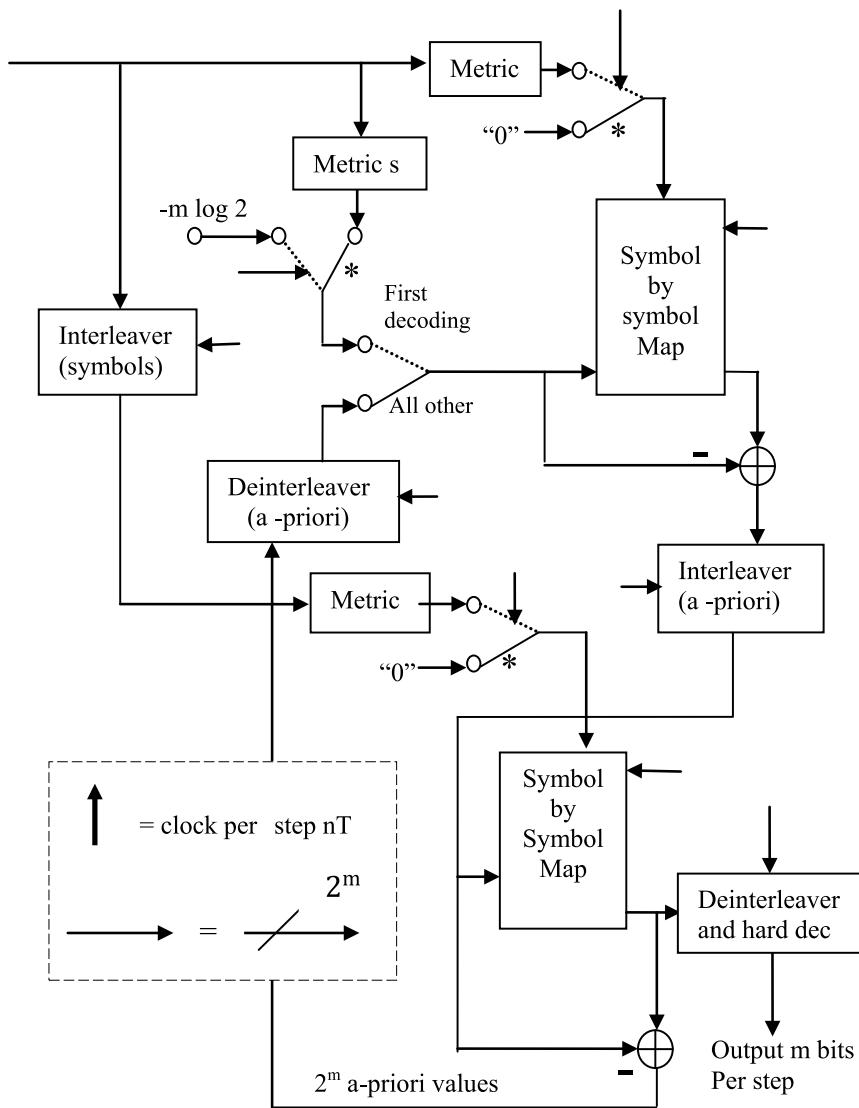


Fig. 7.22 TTCM decoder structure [from Robertson and Worz [8]; © 1998 IEEE.]

noise and the fading that effect the parity components will also affect the corresponding systematic components. Hence, in TTCM, the symbol-based information can be split into two components:

1. The a priori component of the non-binary symbol provided by the alternative decoders.
2. The inseparable extrinsic information as well as the systematic components of the non-binary symbol.

In the first step of TTCM decoding, the received symbols are separated into two different symbols such that upper decoder receives only the symbols encoded by the upper encoder and vice versa for the second decoder. Next, based on log-based BCJR algorithms, each decoder produces its symbol-based probabilities and generates a priori and *extrinsic* information. Next, to make sure that each of the decoder does not receive the same information more than once, the decoders provide the corresponding a posteriori which is subtracted with incoming a priori information. By the random interleavers, the extrinsic information are then interleaved/de-interleaved to become a priori information and made to iterate between them. Then, a posteriori information are de-interleaved from the decoder-2 and uses the hard decision for selecting the maximum a posteriori probability associated with the information word during the final decoding. In the first iteration, the a priori input of the first decoder is initialized with the missing systematic information. Details of the iterative decoder computations are given in the paper by [8].

7.5.3 *Simulation of the BER Performance of the 8-State 8-PSK TTCM in AWGN and Rayleigh Fading Channels*

The schematic for the TTCM is illustrated in Fig. 7.20. The 8-state 8-PSK TCM encoder has shown in Fig. 7.21 is used in this scheme for both the AWGN and Rayleigh fading channels. The source here will be producing some random information bits, which is then encoded by one of the respective encoders and consecutively interleaved by random interleavers. The interleaved bits/symbols are then modulated according to symbol rule for each of the corresponding modulation schemes. The channels discussed here for the coded modulation schemes are the AWGN and Rayleigh-distributed flat fading.

The relationship between the AWGN and Rayleigh fading channel can be expressed as:

$$y_t = \alpha_t x_t + n_t \quad (7.5)$$

where x_t is the transmitted discrete signal and y_t is received signal. α_t is the Rayleigh-distributed fading having an expected squared value of $E(\alpha_t^2)$ and n_t is the complex AWGN having a noise variance of $N_0/2$ per dimension.

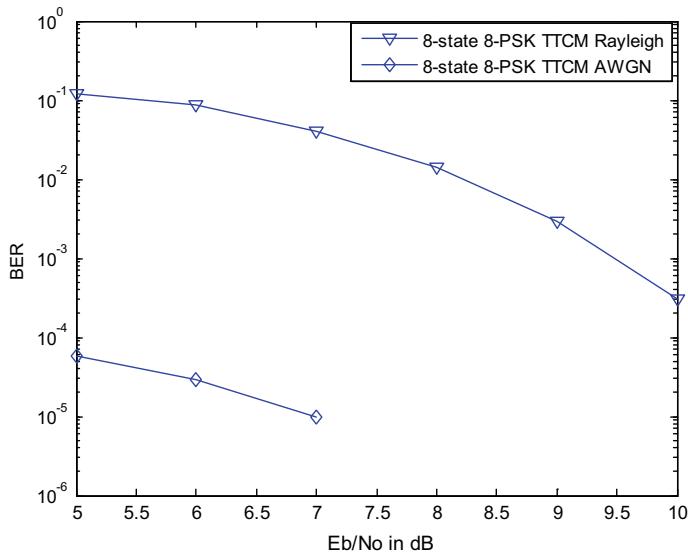


Fig. 7.23 A comparison of the BER performance of the 8-state 8-PSK TTCM in AWGN and Rayleigh fading channel

For an AWGN channel, $\alpha_t = 1$. The receiver side consists of demodulator or demapper followed by a de-interleaver and a TCM or TTCM decoder, which has been explained in the previous chapter. A comparison of the BER performance of 8-state 8-PSK TTCM in the AWGN and Rayleigh fading channel is shown in Fig. 7.23.

A comparison of the BER performance of 8-state 8-PSK TCM and 8-state 8-PSK turbo TCM in AWGN channel is shown in Fig. 7.24.

An additional coding gain of about 1.7 dB has been achieved by use of a turbo TCM compared to the conventional TCM, at error rates in the vicinity of 10^{-4} . This means that turbo TCM achieves a performance close to the Shannon information capacity on an AWGN channel.

7.6 Bit-Interleaved Coded Modulation

Bit-interleaved coded modulation (BICM) was the idea proposed by Zehavi [9] in order to improve the diversity order of TCM scheme. Zehavi's idea was to render the code's diversity equal to that smallest number of different bits by employing the bit-based interleaving as shown in Fig. 7.25. The bit-based interleaving purpose is:

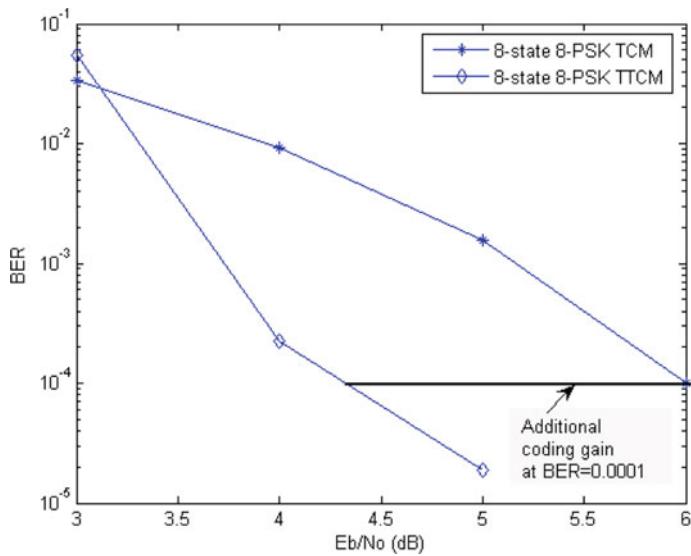


Fig. 7.24 A comparison of the BER performance of 8-state 8-PSK TCM and the 8-state 8-PSK turbo TCM in AWGN channel

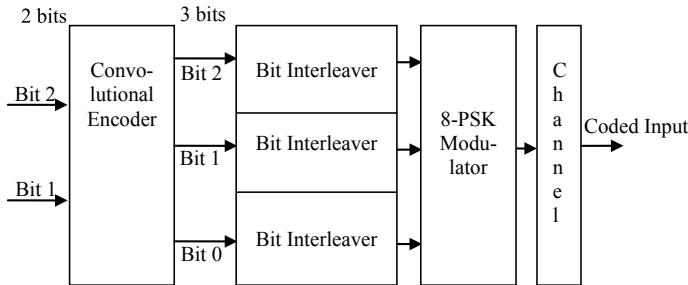


Fig. 7.25 BICM principle

- To maximize the diversity order of the system and to disperse the bursty error introduced by the correlated fading channel.
- To render the bit with respect to the transmitted symbol uncorrelated or independent of each other.

7.6.1 BICM Encoder

The BICM encoder as shown in Fig. 7.26 uses Paaske's non-systematic eight-state code [10] of a rate 2/3 having a free bit-based hamming distance of four for optimum

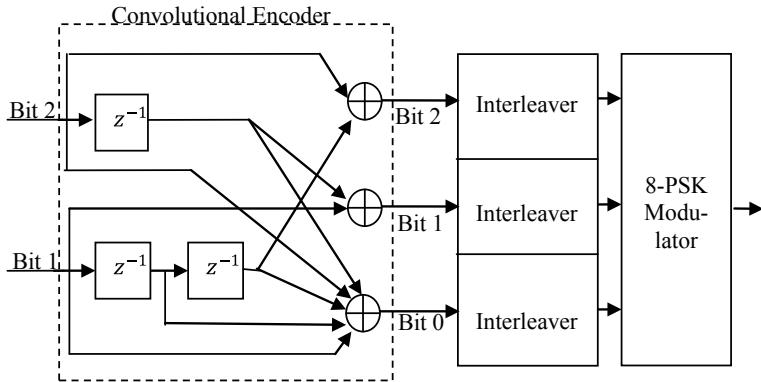


Fig. 7.26 BICM encoder with Paaske's non-systematic convolutional encoder

performance over Rayleigh fading channels. Initially, all the three shift registers contents are set to zero. After the bits are encoded, the each encoded bits will be interleaved by three individual parallel random interleavers of the length equal to each incoming coded bits resulting in a binary vector. These groups of three bits are then mapped to the 8-PSK signal set according to that of Gray mapping.

The content of the three memory elements represents the state of the encoder at an instant. Denoting the state by $S = (s_2 \ s_1 \ s_0)$ as shown in Fig. 7.27, then there are eight possible states S_0 to S_7 .

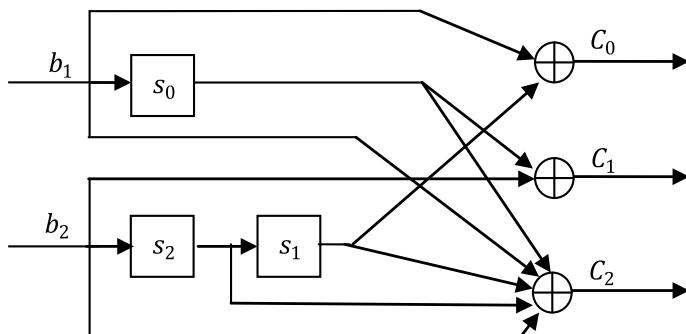


Fig. 7.27 Paaske's non-systematic convolutional encoder

Figure 7.28 shows the trellis diagram with all possible transitions for the encoder shown in Fig. 7.27.

The two-bit information b_1 and b_2 the encoded code word and next states is given by

$$s_0 = b_1; s_1 = s_2; s_2 = b_2 \quad (7.6)$$

$$C_0 = b_1 \oplus s_1; C_1 = b_2 \oplus s_0; C_2 = b_1 \oplus s_0 \oplus s_1 \oplus s_2 \oplus b_2 \quad (7.7)$$

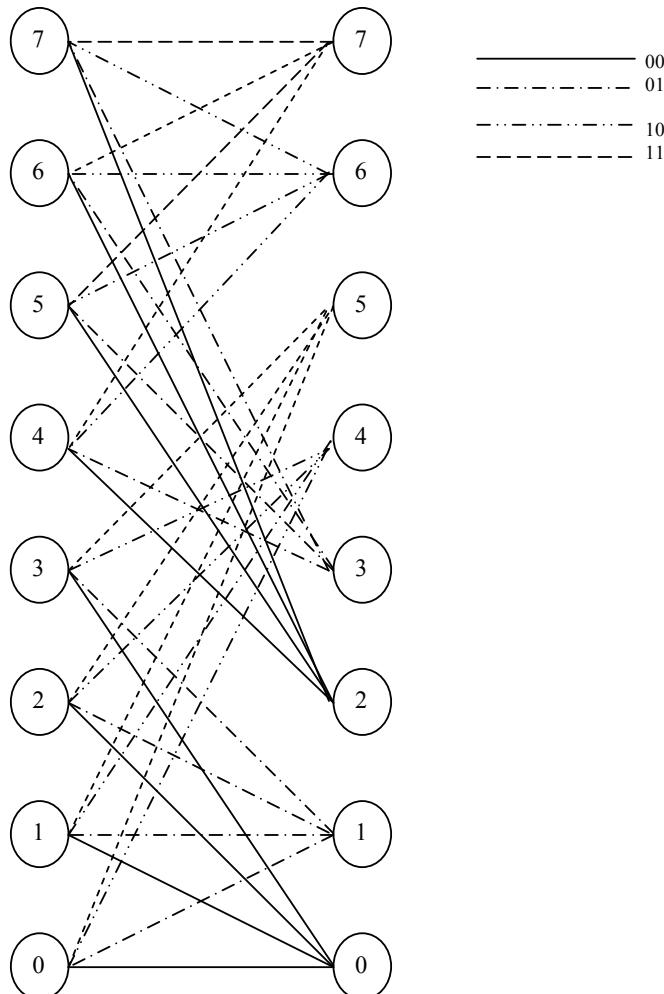
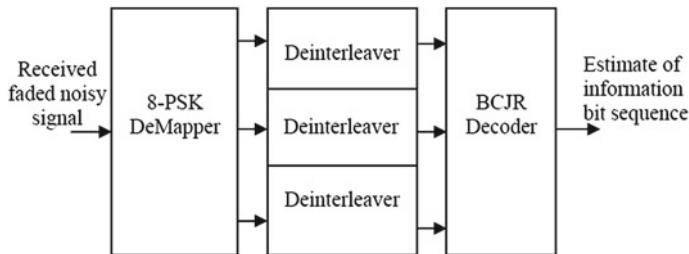


Fig. 7.28 Trellis diagram of the encoder shown in Fig. 7.27

Table 7.1 Code word table for the Paaskes's 8-state convolutional encoder shown in Fig. 7.27

Present states ($s_2 s_1 s_0$)	Information bits							
	00		01		10		11	
Next state	Code word	Next state	Code word	Next state	Code word	Next state	Code word	
000	000	001	101	100	110	101	011	
001	000	110	001	011	100	000	101	
010	000	101	001	000	100	011	101	
011	000	011	001	110	100	101	000	
100	010	100	011	001	110	010	111	
101	010	010	011	111	110	100	111	
110	010	001	011	100	110	111	010	
111	010	111	011	010	110	001	111	

**Fig. 7.29** BICM decoder

for the given set of the information bits b_1 and b_2 , all possible combinations of the code words, present and next states are tabulated in Table 7.1.

7.6.2 BICM Decoder

The BICM decoder is shown in Fig. 7.29. The received faded noisy signal will be demodulated into six-bit metric associated with three-bit positions, each having binary values of 0 and 1, from each received symbol. These bit metrics are then de-interleaved by the three independent bit de-interleavers to form the estimated code words. Then, the BCJR decoder is invoked for decoding these code words to generate the best possible estimate of the original information bits.

7.7 Bit-Interleaved Coded Modulation Using Iterative Decoding

Li and Ritcey [11, 12] have proposed a new scheme of bit-interleaved coded modulation using iterative decoding for further improvement of Zehavi's BICM scheme. The BICM-ID employs set-partitioning signal labeling system as that of Ungerboeck TCM and introduces soft decision feedback from the decoder's output to the demapper/demodulator input to iterate between them. This is advantageous, since it improves the reliability of the soft information passed to the demapper/demodulator at each iteration.

7.7.1 BICM-ID Encoder and Decoder

The BICM-ID's encoder is similar to that BICM encoder explained in Fig. 7.26. The BICM-ID's decoder is almost similar to that of the BICM's encoder except that the iterative process is used to achieve global optimum through a step-by-step local search.

The Fig. 7.30 shows the BICM-ID decoder. At the initial step, the received signal r is demodulated and generated the extrinsic information of the coded bits $P(\bar{c}; O)$ which is interleaved by corresponding de-interleavers to become the a priori information $P(c; I)$ to the log-based BCJR decoders to generate a posteriori bit probabilities for the information and the coded word.

On the second pass, the extrinsic, a posteriori vectors are interleaved as a priori information to the demodulator assuming that all the bits are independent of each other (by a design of a good interleaver) and will again iterate the above said steps until the final step is reached. The total a posteriori probabilities of the information bits can be computed to make the hard decisions at the output of the decoder after the each iteration.

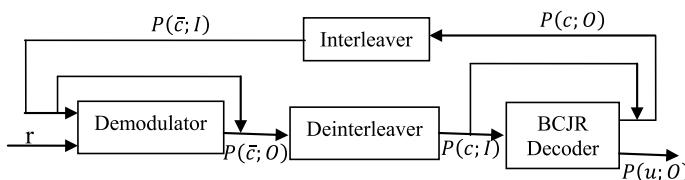


Fig. 7.30 BICM-ID decoder

The SISO channel decoder uses the MAP algorithm similar to decoding of Turbo codes; here, the demodulator and the channel decoder exchange the extrinsic information of the coded bits $P(\bar{c}; O)$ and $P(c; O)$ through an iterative process. After being interleaved, $P(\bar{c}; O)$ and $P(c; O)$ become a priori information $P(c; I)$ and $P(\bar{c}; I)$ at the input of the BCJR decoder and the demodulator, respectively.

7.7.2 *Simulation of the BER Performance of 8-State 8-PSK BICM and BICM-ID in AWGN and Rayleigh Fading Channels*

Simulations are carried out for BICM and BICM-ID with the 8-state 8-PSK encoders. The interleavers used here are three parallel independent random interleavers. The BER performance of BICM in an AWGN channel for three parallel 512 bits interleavers and three parallel 3000 bits interleavers is shown in Fig. 7.31. From Fig. 7.31, it is observed that the BER performance of BICM does not significantly depend on the frame length. The BER performance of BICM and BICM-ID (with 2 iterations

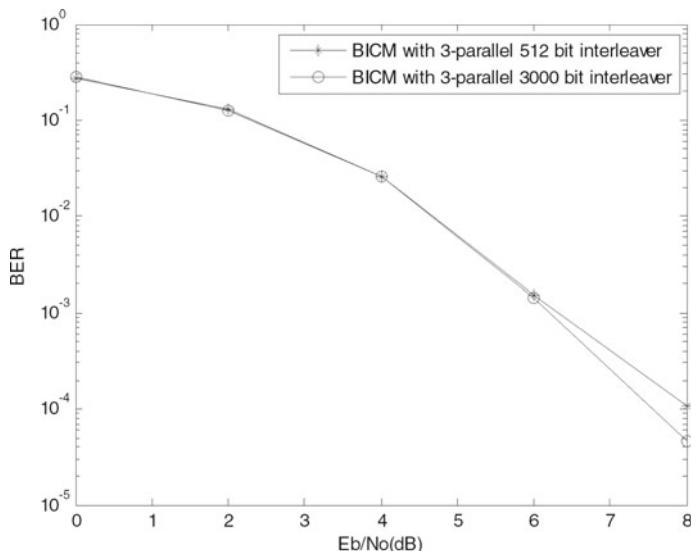


Fig. 7.31 BER performance of BICM in an AWGN channel

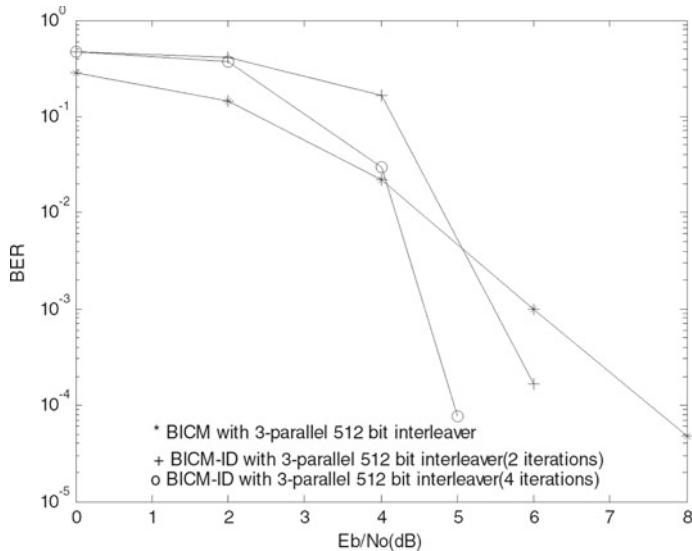


Fig. 7.32 BER performances of BICM and BICM-ID in AWGN channel

and 4 iterations) in an AWGN channel for three parallel 512 bits interleavers is shown in Fig. 7.32. It is seen from Fig. 7.32 that the BER performance improves with the increased number of iterations. The BER performance of BICM and BICM-ID (with 4 iterations) in Rayleigh fading channel for three parallel 1000 bits interleavers is shown in Fig. 7.33.

7.8 Problems

1. Obtain the set partition for 16 QAM.
2. Draw the trellis diagram for the following 16-state 8-PSK TCM encoder and find the asymptotic coding gain.

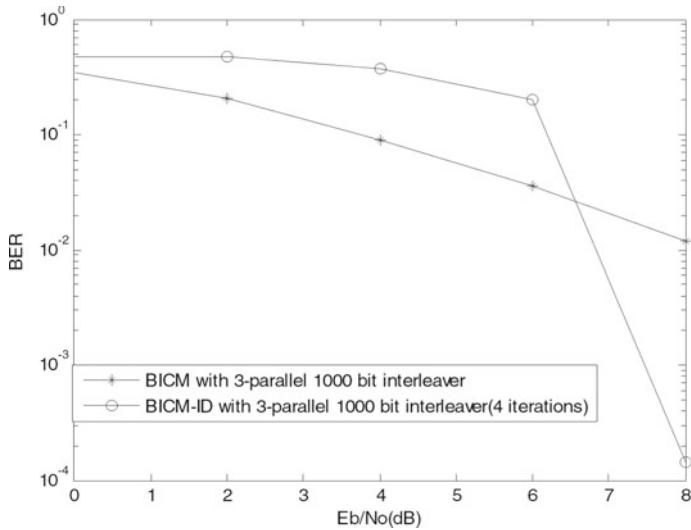
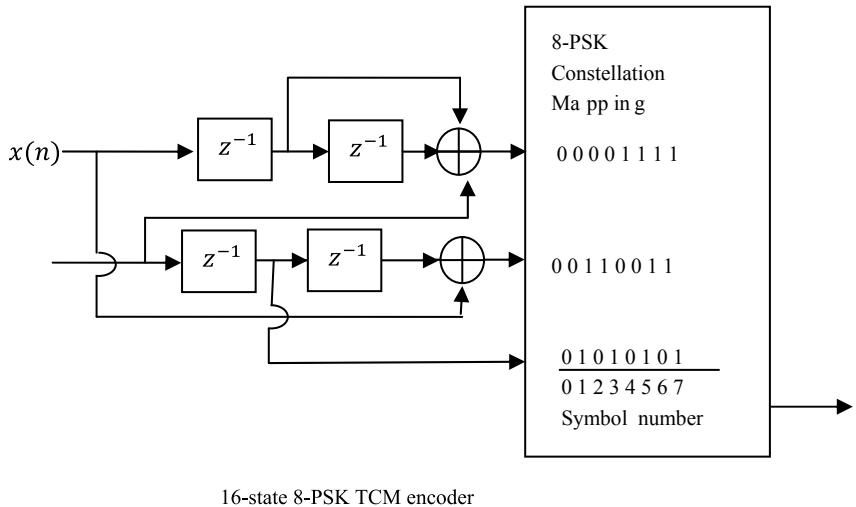


Fig. 7.33 BER performance of BICM and BICM-ID in Rayleigh fading channel



16-state 8-PSK TTCM encoder

3. Compute the output symbols of the 8-state 8-PSK TTCM encoder shown in Figure for the input information bit pairs {00, 01, 11, 10, 00, 11}. Let the interleaver be {3, 6, 5, 2, 1, 4}.
4. Construct the schematic diagram for the 8-state 16-QAM TTCM encoder.

Appendix A

```

function [Codeword,Nextstate, Previousstate]= genpoly(k,n,L)
%Init the shift register for the TCM
L=L+1; cpoly=getgenpoly(k,L-1);
H=zeros(k+1,L); K=zeros(1,k+1); D=zeros(1,L);
Codeword=zeros(2^(L-1),2^k); Nextstate=zeros(2^(L-1),2^k);
Previousstate=zeros(2^(L-1),2^k); h=0;
for i=1:L
    h=mod(h,3);
    if (h==0)
        for m=1:k+1
            K(1,m)=mod(cpoly(1,m),10);
        cpoly(1,m)=floor(cpoly(1,m)/10);
        end; end
        h=h+1;
        for m=1:k+1
            H(m,i)=mod(K(1,m),2); K(1,m)=floor(K(1,m)/2); end;end
    if (H(1,1)==1)
        error('TCM: the feedback poly is unacceptable');
    end
    for s=1:2^(L-1)
        h=s-1;
        for i=1:L-1
            D(1,i) = mod(h,2); h = floor(h/2); end
            for m=1:2^k
                h=m-1;
                for i=2:k+1
                    K(1,i) = mod(h,2); h=floor(h/2); end
                    h=D(1,1);
                    for i=2:k
                        h = mod((h+ K(1,i)*H(i,1)),2); end
                        Codeword(s,m) = 2^(m-1) + h;
                        K(1,1) = h; c = 1;Nextstate(s,m) = 0; %///* compute new state */
                        for j=1:L-1
                            %/* bit from previous reg. */
                            if(j < (L-1))
                                h=D(1,j+1);
                            else
                                h=0; end
                            for i=1:k+1 % /* input and feedback bits */
                                h = mod(h +K(1,i)*H(i,j+1),2); end
                                Nextstate(s,m)=Nextstate(s,m)+(h*c); %/* add to state */
                                c=c*2; end; end; end
                    for i=1:2^(L-1) %/Compute Pevious State/
                        for j= 1:2^k
                            Previousstate((Nextstate(i,j)+1),j)=(i-1); end; end

```

```

function [GenPoly] = getgenpoly(k,L)
switch(k)
case 1
    switch (L)
        case 1
            GenPoly(1,1) = 1; GenPoly(1,2) = 2;
        case 3
            GenPoly(1,1) = 13; GenPoly(1,2) = 6;
        case 4
            GenPoly(1,1) = 23; GenPoly(1,2) = 6;
        case 6
            GenPoly(1,1) = 117; GenPoly(1,2) = 26;
        case 7
            GenPoly(1,1) = 217; GenPoly(1,2) = 110;
        case 8
            GenPoly(1,1) = 427; GenPoly(1,2) = 230;
        case 9
            GenPoly(1,1) = 1017; GenPoly(1,2) = 120;
        otherwise
            error('no generator for such code yet for 4QAM');
    end
case 2
    switch(L)
        case 3
            GenPoly(1,1) = 11; GenPoly(1,2) = 2; GenPoly(1,3) = 4;
        case 4
            GenPoly(1,1) = 23; GenPoly(1,2) = 2; GenPoly(1,3) = 10;
        case 6
            GenPoly(1,1) = 103; GenPoly(1,2) = 30; GenPoly(1,3) = 66;
        case 7
            GenPoly(1,1) = 277; GenPoly(1,2) = 54; GenPoly(1,3) = 122;
        case 8
            GenPoly(1,1) = 435; GenPoly(1,2) = 72; GenPoly(1,3) = 130;
        otherwise
            error('no generator for such code yet for 8 PSK');
    end
end
end

```

```

function [smap,bps,nl] =PSKmodSP(varargin)
n=varargin{1};
nl=bitshift(1,n);
M=nl;
smap=zeros(1,nl);
bps=n;
for j=1:M
    smap(1,j)=complex((cos(2*pi*(j-1)/M)),(sin(2*pi*(j-1)/M)));
end

```

```

function [ symbols ] = bits2symbol(word_length,block_lenght,bits_seq)
    N=block_lenght*word_length;
    symbols=zeros(1,block_lenght);
    if (N ~= length(bits_seq))
        error('bits_seq_to_symbol: check bits_seq.length()');
    end
    k=1;
    for j=1:block_lenght
        for i=1:word_length
            symbols(1,j)= symbols(1,j)+ (bits_seq(1,(k))*bitshift(1,(i-1)));
            k=k+1;
        end
    end
end

```

```

function [ Output_symbols,Tx_signals ] = tcmenc(symbols,Codeword,
Nextstate,Smap)
    Output_symbols=zeros(1,length(symbols));
    s=1;
    for i=1:length(Output_symbols)
        m=symbols(1,i)+1;
        Output_symbols(1,i)=Codeword(s,m);
        s=Nextstate(s,m)+1;
    end
    Tx_signals=zeros(1,length(Output_symbols));
    for i=1:length(Tx_signals)
        Tx_signals(1,i)=Smap(Output_symbols(1,i)+1);
    end
end

```

```

function [ Pr ] = demodsymbols(varargin)
    global M N smap nl nis;
    received_signals= varargin{1};
    sigma= varargin{2};
    %Channel Matrix
    Pr=zeros(N,2*M);
    for k=1:nis
        for i=1:nl
            dist=hypot((real(received_signals(1,k))- real(smap(1,i))), (imag
            (received_signals(1,k))- imag(smap(1,i))));
            Pr(k,i)=-(dist*dist)/(sigma);
        end
    end
end

```

```

function [ b_decoded_bits ] = bitsdecode(Pr)
    global M N S Cw Nes Prs MINF Interleave_mode_in;
    MINF=-100000; %// define Minimum Log probability (-infinity)
    Apr=zeros(N,M); Apo=zeros(N,M); OPr=zeros(N,2*M); Ip1=zeros(S,M,N);
    if (strcmp(Interleave_mode_in,'ON'))
        y=de_interleave(Pr); Pr=y; end
    for j=1:N
        for m=1:M
            Apr(j,m)=-log(M); for i=1:S
                Ip1(i,j,m)=Pr(j,Cw(i,m)+1); end; end; end
        Alpha=zeros(N+1,S+1);Beta=zeros(N+1,S+1); for i=2:S
            Alpha(1,i)=MINF; /* compute Alpha */ end
        for k=2:(N+1)
            max=MINF; for i=1:S
                Alpha(k,i)=jacobianlog(Alpha(k-1,Prs(i,1)+1) + Ip1(Prs(i,1)+1,1,(k-1))+Apr(k-1,1),Alpha(k-1,Prs(i,2)+1) + Ip1(Prs(i,2)+1,2,(k-1))+Apr(k-1,2));
                for m=3:M
                    Alpha(k,i)=jacobianlog(Alpha(k,i),Alpha((k-1),Prs(i,m)+1)+Ip1(Prs(i,m)+1,m,(k-1))+Apr(k-1,m)); end
                    if (max < Alpha(k,i))
                        max=Alpha(k,i); end; end
            for i=1:S
                Alpha(k,i)=Alpha(k,i)-max;
            end end for i=1:S
            Beta(N+1,i)=0; /* compute beta */
        end for k=N:-1:1
        max=MINF;for i=1:S
            Beta(k,i)=jacobianlog(Beta(k+1,Nes(i,1)+1) +
Ip1(i,1,k)+Apr(k,1),Beta(k+1,Nes(i,2)+1) + Ip1(i,2,k)+Apr(k,2));
            for m=3:M
                Beta(k,i)=jacobianlog(Beta(k,i),Beta(k+1,Nes(i,m)+1)+Ip1(i,m,k)+Apr(k,m));
            end
            if (max < Beta(k,i))
                max=Beta(k,i); end; end
            for i=1:S
                Beta(k,i)=Beta(k,i)-max;
            end end /* compute apo */
        for k=1:N
            max=MINF; max_QPr=MINF;
            for m=1:(2*M)
                OPr(k,m)=MINF;
            End for m=1:M
                Apo(k,m)=MINF; for i=1:S
                    abc=Alpha(k,Prs(i,m)+1)+ Beta(k+1,i)+Ip1(Prs(i,m)+1,m,k);
                Apo(k,m)=jacobianlog(Apo(k,m),abc);
        
```

```

OPr(k,Cw(Prs(i,m)+1,m)+1)=jacobianlog(OPr(k,Cw(Prs(i,m)+1,m)+1),abc+Apr(k,m
));
end
Apo(k,m)=Apo(k,m)+Apo(k,m); if (max < Apo(k,m))
    max=Apo(k,m);
end end
for m=1:M
    Apo(k,m)=Apo(k,m)- max;
end end
decoded_symbols=decode_symbols(Apo)
b_decoded_bits=symbol2bits(decoded_symbols); end

```

```

function [ r ] = jacobianlog( x,y )
%/*----- jacobian logarithm -----*/
if (x > y)
    r=x + log (1 + exp(y-x));
else
    r=y + log (1 + exp(x-y));
end
end

```

```

function [ output_symbols ] = decode_symbols(Apo)
global N M
output_symbols=zeros(1,N);
for k=1:N
    i=0;
    max=Apo(k,1);
    for m=2:M
        if (Apo(k,m) > max)
            max=Apo(k,m);
            i=m-1;
        end
    end
    output_symbols(1,k)=i;
end
end

```

```

function [ bits ] = symbol2bits( symbols)
    global n;
    N=n*length(symbols);
    bits=zeros(1,N);
    if(N ~= length(bits))
        msgbox(N,length(bits))
    end
    h=1;
    for j=1:length(symbols)
        for i=1:n
            bits(h)=bitand(bitshift(symbols(1,j),-(i-1)),1);
            h=h+1;
        end
    end
end

```

References

1. G. Ungerboeck, Channel coding with multilevel/phase signals. *IEEE Trans. Inf. Theory* **28**, 55–67 (1982)
2. G.D. Forney, R.G. Gallager, G.R. Lang, F.M. Longstaff, S.U. Qureshi, Efficient modulation for band-limited channels. *IEEE Trans. Sel. Areas. Commun.* **SAC-2**, 632–647 (1984)
3. L.F. Wei, Rotationally invariant convolutional channel coding with expanded signal space-part I: 180 degrees. *IEEE Trans. Sel. Areas. Commun.* **SAC-2**, 659–672 (1984)
4. L.F. Wei, Rotationally invariant convolutional channel coding with expanded signal space-part 11: nonlinear codes. *IEEE Trans. Sel. Areas. Commun.* **SAC-2**, 672–686 (1984)
5. A.R. Calderbank, J.E. Mazo, A new description of trellis codes. *IEEE Trans. Inf. Theory* **IT-30**, 784–791 (1984)
6. CCITT Study Group XVII, Recommendation V.32 for a family of 2-wire, duplex modems operating on the general switched telephone network and on leased telephone-type circuits. Document AP VIII-43-E, May 1984
7. CCITT Study Group XVII, Draft recommendation V.33 for 14400 bits per second modem standardized for use on point-to-point 4-wire leased telephone-type circuits. Circular No. 12, COM XVII/YS, Geneva, 17, May 1985
8. P. Robertson, T. Worz, Bandwidth-efficient turbo trellis-coded modulation using punctured component codes. *IEEE J. Sel. Areas Commun.* **16**, 206–218 (1998)
9. E. Zehavi, 8-PSK trellis codes for rayleigh fading channel. *IEEE. Trans. Commun.* **40**, 873–883 (1992)
10. S. Lin, D. Constello Jr., *Error Control Coding: Fundamentals and Applications* (Prentice Hall, Englewood Cliff, NJ, USA, 1982), ISBN: 013283796X. October 1982
11. X. Li, J.A. Ritcey, Bit interleaved coded modulation with iterative decoding. *IEEE Commun. Lett.* **1**. November 1997
12. X. Li, J.A. Ritcey, Trellis coded modulation with bit interleaving and iterative decoding. *IEEE J. Sel. Areas Commun.* **17**. April 1999

Chapter 8

Low Density Parity Check Codes



Low density parity check (LDPC) codes are forward error-correction codes, invented by Robert Gallager in his MIT Ph.D. dissertation, 1960. The LDPC codes are ignored for long time due to their high computational complexity and domination of highly structured algebraic block and convolutional codes for forward error correction. A number of researchers produced new irregular LDPC codes which are known as new generalizations of Gallager's LDPC codes that outperform the best turbo codes with certain practical advantages. LDPC codes have already been adopted in satellite-based digital video broadcasting and long-haul optical communication standards. This chapter discusses LDPC Code Properties, construction of parity check matrix for regular and irregular LDPC codes, efficient Encoding and Decoding of LDPC Codes, performance analysis of LDPC Codes.

8.1 LDPC Code Properties

Low Density Parity Check (LDPC) code is a linear error-correcting code that has a parity check matrix H , which is sparse i.e. with less nonzero elements in each row and column. LDPC codes can be categorized into regular and irregular LDPC codes. When the parity-check matrix $H_{(n-k) \times n}$ has the same number w_c of ones in each column and the same number w_r of ones in each row, the code is a regular (w_c, w_r). The original Gallager codes are regular binary LDPC codes. The size of H is usually very large, but the density of nonzero element is very low. LDPC code of length n , or denoted as an (n, w_c, w_r) LDPC code. Thus, each information bit is involved with w_c parity checks, and each parity-check bit is involved with w_r information bits. For a regular code, we have $(n - k)w_r = nw_c$ thus $w_c < w_r$. If all rows are linearly independent, the code rate is $\frac{(w_r - w_c)}{w_r}$, otherwise it is k/n . Typically, $w_c \geq 3$. A parity check matrix with minimum column weight w_c will have a minimum distance $d_{\min} \geq w_c + 1$. When $w_c \geq 3$, there is at least one LDPC code whose minimum

distance d_{\min} grows linearly with the block length n [1]; thus a longer code length yields a better coding gain. Most regular LDPC codes are constructed with w_c and w_r on the order of 3 or 4.

8.2 Construction of Parity Check Matrix H

8.2.1 *Gallager Method for Random Construction of H for Regular Codes*

In this method, the transpose of regular (n, w_c, w_r) parity check matrix H has the form

$$H^T = [H_1^T, H_2^T, \dots, H_{w_c}^T] \quad (8.1)$$

The matrix H_1 has n columns and n/w_r rows. The H_1 contains a single 1 in each column and contains 1s in its i th row from column $(i - 1)w_r + 1$ to column $i w_r$. Permuting randomly the columns of H_1 with equal probability, the matrices H_2 to H_{w_c} are obtained.

The parity check matrix for $(n = 20, w_c = 3, w_r = 4)$ code constructed by Gallager [1] is given as

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (8.2)$$

The following MATLAB program can be used to generate Gallager regular parity check matrix H with different code rates.

Program 8.1 MATLAB program to generate Gallager regular parity check matrix.

```
% MATLAB program to generate Gallager regular parity check matrix H
clear all; clc;
c = input(' enter the Length of parity check bits = ');
n = input(' enter the number of bits in the codeword= ');
wc = input(' enter the number of ones in each column= ');
wr=(n.*wc)./c; %the number of ones in each row
H=zeros(c,n); %generate the empty matrix and start assigning the 1s
j=1;
jj=wr;
for i = 1:c./wc
    H(i,j:jj)=1;
    j=j+wr;
    jj=(i.*wr)+wr;
end
for i=1:wc-1
    for ii=1:n
        colind = (round(rand(1) * (n-1)))+1;
        rCol=H(1:c./wc,colind);
        H((i.*(c./wc))+1 : ((i.*(c./wc))+1 + (c./wc)-1),ii )=rCol;
    end
end
```

8.2.2 Algebraic Construction of H for Regular Codes

The construction of the parity check matrix H using algebraic construction as follows [2, 3]. Consider an identity matrix I_a where $a > (w_c - 1)(w_r - 1)$ and obtain the following matrix by cyclically shifting the rows of the identity matrix I_a by one position to the right.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 1 \\ 1 & 0 & 0 & 0 & \dots & 0 \end{bmatrix} \quad (8.3)$$

Defining $A^0 = I_a$ the parity check matrix H can be constructed as

$$H = \begin{bmatrix} A^0 & A^0 & A^0 & \dots & A^0 \\ A^0 & A^1 & A^2 & \dots & A^{(w_r-1)} \\ A^0 & A^2 & A^4 & \dots & A^{2(w_r-1)} \\ & & & \ddots & \\ A^0 & A^{(w_c-1)} & A^{2(w_c-1)} & \dots & A^{(w_c-1)(w_r-1)} \end{bmatrix} \quad (8.4)$$

The constructed H matrix has $w_c a$ rows and $w_r a$ columns, and it is of a regular $(w_r a, w_c, w_r)$ having the same number of w_r ones in each row and the same number of w_c ones in each column. It is four cycle free construction. The algebraic LDPC codes are easier for decoding than random codes. For intermediate n , well designed algebraic codes yields a low BER [4, 5].

Example 8.1 Construct H matrix with $w_c = 2$ and $w_r = 3$ using algebraic construction method.

Solution Since $(w_c - 1)(w_r - 1) = 2$

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

$$H = \begin{bmatrix} A^0 & A^0 & A^0 \\ A^0 & A^1 & A^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

8.2.3 Random Construction of \mathbf{H} for Irregular Codes

In the random construction of the parity check matrix H , the matrix is filled with ones and zeros randomly satisfying LDPC properties. The following MATLAB program generates rate 1/2 irregular parity check matrix H with ones distributed uniformly at random within the column.

Program 8.2 MATLAB program to generate rate 1/2 irregular parity check matrix H .

```
%MATLAB program to generate rate 1/2 irregular parity check matrix H
% with 1s distributed uniformly at random within column
clear all;clc;
p = input(' enter the Length of parity check bits = ');
n = input(' enter the number of bits in the codeword= ');
col_ones= input(' enter the number of ones per column= ');
for i = 1:n
    ones_col(:, i) = randperm(p)';
end
r = reshape(ones_col(1:col_ones, :), n*col_ones, 1);
temp = repmat([1:n], col_ones, 1);
c = reshape(temp, n*col_ones, 1);
H= full(sparse(r, c, 1, p, n));% Creates sparse matrix H
for i = 1:p
    cr = randperm(n);
    if length(find(r == i)) == 0
        H(i, cr(1)) = 1;% adds two 1s if row has no 1
        H(i,cr(2)) = 1;
    elseif length(find(r == i)) == 1
        H(i, cr(1)) = 1;% adds one 1 if row has only one 1
    end
end % for i
```

An example of parity check matrix for irregular LDPC code is

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \quad (8.5)$$

8.3 Representation of Parity Check Matrix Using Tanner Graphs

The Tanner graph of the parity check matrix H is a bipartite graph. It has bit nodes or variable nodes (VN) equal to the number of columns of H , and check nodes (CN) equal to the number of rows of H . If $H_{ji} = 1$, i.e. if variable i participates in the j th parity-check constraint, then check node j is connected to variable node i .

Example 8.2 Construct Tanner graph for the following parity check matrix

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Solution The H matrix has 10 columns and 5 rows. Hence, the associated tanner graph with 10 bit nodes and 5 check nodes is shown in Fig. 8.1.

8.3.1 Cycles of Tanner Graph

Consider the following parity-check matrix

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \quad (8.6)$$

The Tanner graph of the H matrix is shown in Fig. 8.2. A sequence of connected nodes starting and ending at the same node with no node more than once is a cycle of a Tanner graph. The number of edges in a cycle is called cycle length and the smallest size of the cycle in a graph represents the girth of the graph. Cycles of length 4 situations arise where pairs of rows share 1s in a particular pair of columns of the above H matrix. A cycle of length 4 is shown in bold in Fig. 8.2.

The minimum lower bound distance for four cycle free (w_c, w_r) regular LDPC code parity check matrix with girth g , is given by [6]

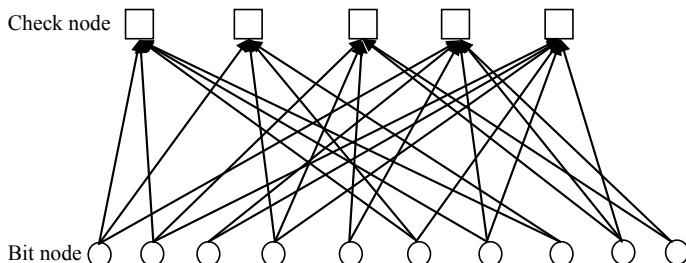


Fig. 8.1 Tanner graph of H matrix of Example 8.2

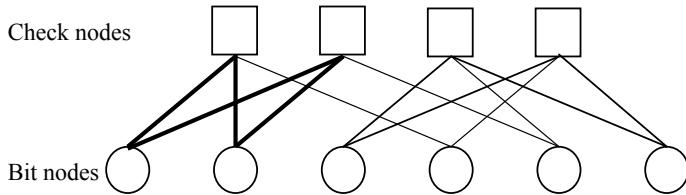


Fig. 8.2 A Tanner graph with a cycle of length 4

$$d_{min} \geq \begin{cases} 1 + w_c + w_c(w_c - 1) + w_c(w_c - 1)^2 + \dots + w_c(w_c - 1)^{(g-6)/4} & \text{for odd } g/2 \\ 1 + w_c + w_c(w_c - 1) + w_c(w_c - 1)^2 + \dots + w_c(w_c - 1)^{\frac{g-8}{4}} & \text{otherwise} \end{cases} \quad (8.7)$$

Thus the minimum distance can be increased by increasing the girth or the column weight.

8.3.2 *Detection and Removal of Girth 4 of a Parity Check Matrix*

If the Tanner graph of a parity check matrix contains no loops then this decoding is quickly computable. Unfortunately LDPCs have loopy graphs, and so the algorithm needs to be repeatedly iterated until it converges to a solution. The effect of girth on the performance of LDPC codes can be reduced by choosing the codes having Tanner graphs with longer girths. However, longer girths are not helpful for finite length codes. A girth of 6 is sufficient and hence the removal of girth 4 is a required. A lemma in [7] states that the H matrix has no girth 4, if and only if all the entries of the matrix $[H^T H]$ are 1s except the diagonal line.

A standard approach [8] is to search the parity-check matrix H forming a rectangle of four 1s in the matrix. Eliminating the rectangle by reshuffling some elements around while preserving the other relevant properties of the matrix is equivalent to removing a girth 4 from the Tanner graph.

The detection and removal of girth 4 is illustrated through the following numerical example using MATLAB.

Example 8.3 Consider the following $(10, 3, 6)$ regular parity check matrix

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

The following MATLAB program can be used for detection and removal of girth of the given H matrix

Program 8.3 MATLAB program for detection and removal of girth 4 of a given parity check matrix H .

```

clear all;clc;
H=[1 1 1 1 0 1 1 0 0 0;0 0 1 1 1 1 1 0 0;0 1 0 1 0 1 0 1 1 1;1 0 1 0 1 0 0 1 1
1;1 1 0 0 1 0 1 0 1];
[row col]=size(H);
Gt=H'*H;
for i=1:row
Gt(i,i)=0;
end
figure(1),
mesh(Gt)% shows presence or absence of girth 4
d=find(Gt==2);
if(d~=0)
%removal of girth 4
for i = 1:row
for j = (i + 1):row
sp = and(H(i, :), H(j, :));
csp = find(sp);
cl = length(csp);
if cl > 1
if length(find(H(i, :))) < length(find(H(j, :)))
for cp = 1:cl - 1
H(j, csp (cp )) = 0;
end
else
for cp = 1:cl - 1
H(i, csp (cp )) = 0;
end
end % if
end % if
end % for j
end % for i
end %for if
figure(2),mesh(H) % shows girth 4 free H

```

The results obtained from the above MATLAB program are shown in Figs. 8.3 and 8.4.

From Fig. 8.3, it is observed that all the entries of the matrix $[H^T H]$ except diagonal line are not 1s. Hence, the given H matrix has girth 4, whereas Fig. 8.4 shows girth 4 free H .

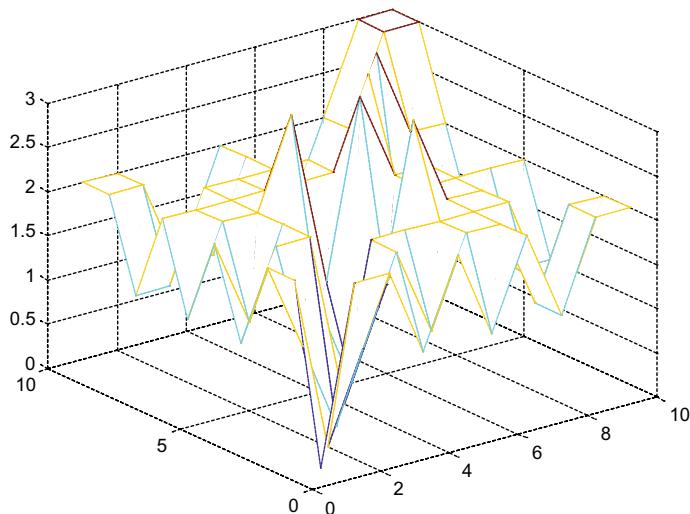


Fig. 8.3 Entries of H with girth 4

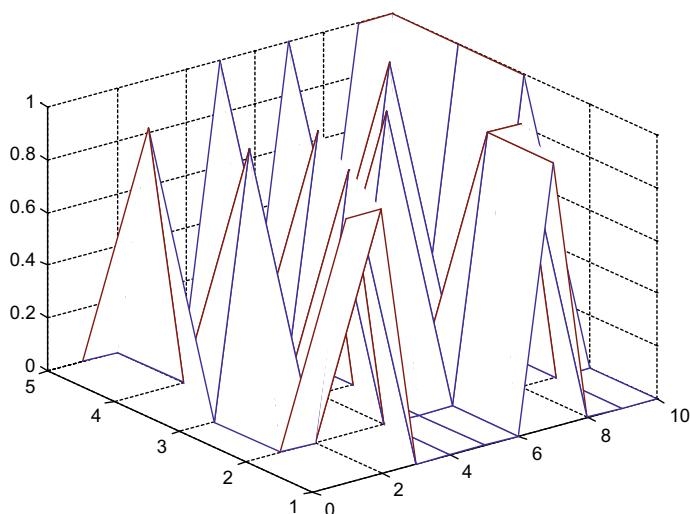


Fig. 8.4 Entries of girth 4 free H

8.4 LDPC Encoding

8.4.1 Preprocessing Method

For coding purposes, we may derive a generator matrix G from the parity check matrix H for LDPC codes by means of Gaussian elimination in modulo-2 arithmetic. Since the matrix G is generated once for a parity check matrix, it is usable in all encoding of messages. As such this method can be viewed as the preprocessing method.

1-by- n code vector c is first partitioned as

$$C = [b : m] \quad (8.8)$$

where m is k by 1 message vector, and b is the $n - k$ by 1 parity vector correspondingly, the parity check matrix H is partitioned as

$$H^T = \begin{bmatrix} H_1 \\ \dots \\ H_2 \end{bmatrix} \quad (8.9)$$

where H_1 is a square matrix of dimensions $(n - k) \times (n - k)$, and H_2 is a rectangular matrix of dimensions $k \times (n - k)$ transposition symbolized by the superscript T is used in the partitioning of matrix H or convenience of representation.

Imposing the constraint $CH^T = 0$.

We may write

$$[b : m] \begin{bmatrix} H_1 \\ \dots \\ H_2 \end{bmatrix} = 0 \quad (8.10)$$

or equivalently,

$$bH_1 + mH_2 = 0 \quad (8.11)$$

The vectors m and b are related by

$$b = mP \quad (8.12)$$

where P is the coefficient matrix. For any nonzero message vector m , the coefficient matrix of LDPC codes satisfies the condition

$$PH_1 + H_2 = 0 \quad (8.13)$$

Which holds for all nonzero message vectors and, in particular, in the form $[0 \dots 0 1 0 \dots 0]$ that will isolate individual rows of the generator matrix. Solving Eq. (8.13) for matrix P , we get

$$P = H_2 H_1^{-1} \quad (8.14)$$

where H_1^{-1} is the inverse matrix of H_1 , which is naturally defined in modulo-2 arithmetic. Finally, the generator matrix of LDPC codes is defined by

$$G = [P : I_k] = [H_2 H_1^{-1} : I_k] \quad (8.15)$$

where I_k is the k by k identity matrix. The codeword can be generated as

$$C = mG \quad (8.16)$$

Example 8.4 Construct generator matrix G for the following (10, 3, 5) regular parity check matrix.

$$\left[\begin{array}{cccccc|ccc} 1 & 1 & 0 & 1 & 0 & 1 & \vdots & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & \vdots & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & \vdots & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & \vdots & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & \vdots & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & \vdots & 1 & 1 & 1 & 1 \end{array} \right]$$

Solution

$$H = \underbrace{\left[\begin{array}{cccccc|ccc} \vdots & & & & & & & & & & \\ 1 & 1 & 0 & 1 & 0 & 1 & \vdots & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & \vdots & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & \vdots & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & \vdots & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & \vdots & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & \vdots & 1 & 1 & 1 & 1 \end{array} \right]}_{H_1^T} : \underbrace{\vdots}_{\vdots} : \underbrace{\left[\begin{array}{ccc} \vdots & & \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{array} \right]}_{H_2^T}$$

$$H_1 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \quad H_2 = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Letting $mH_2 = u$, the following relation can be written from Eq. (8.11)

$$\begin{bmatrix} b_0 & b_1 & b_2 & b_3 & b_4 & b_5 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} u_0 & u_1 & u_2 & u_3 & u_4 & u_5 \end{bmatrix}$$

The above relation between b and u leads to the following equations

$$\begin{aligned} b_0 + b_1 + b_3 + b_5 &= u_0 \\ b_1 + b_2 + b_4 &= u_1 \\ b_0 + b_4 + b_5 &= u_2 \\ b_1 + b_2 + b_3 + b_5 &= u_3 \\ b_0 + b_2 + b_4 &= u_4 \\ b_3 &= u_5 \end{aligned}$$

Solving the above equations, using modulo-2 arithmetic, we obtain

$$\begin{aligned} b_0 &= u_1 + u_2 + u_3 + u_5 \\ b_1 &= u_2 + u_3 + u_4 + u_5 \\ b_2 &= u_0 + u_1 + u_2 + u_5 \\ b_3 &= u_5 \\ b_4 &= u_0 + u_3 + u_4 \\ b_5 &= u_0 + u_1 + u_4 + u_5 \end{aligned}$$

Since $b = uH_1^{-1}$, the above equations can be write in matrix form ads

$$b = [u] \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Thus,

$$\begin{aligned}
 H_1^{-1} &= \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \\
 H_2 H_1^{-1} &= \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} \\
 &= \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}
 \end{aligned}$$

The generator matrix

$$\begin{aligned}
 G &= [H_2 H_1^{-1} \ I_k] \\
 &= \underbrace{\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}}_{H_2 H_1^{-1}} \ : \ \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{I_k}
 \end{aligned}$$

Example 8.5 Construct LDPC code word for the following parity check matrix with the message vector $m = [1 \ 0 \ 0 \ 0 \ 1]$.

$$H = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Solution The parity check matrix H is of the order 5×10 . We know that $H^T = \begin{bmatrix} H_1 \\ H_2 \end{bmatrix}$, then

$$H^T = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$H_1 = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix} \quad \text{and} \quad H_2 = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Letting $mH_2 = u$, the following relation can be written from Eq. (8.11)

$$\begin{bmatrix} b_0 & b_1 & b_2 & b_3 & b_4 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} u_0 & u_1 & u_2 & u_3 & u_4 \end{bmatrix}$$

The above relation between b and u leads to the following equations

$$\begin{aligned} b_0 + b_1 + b_4 &= u_0 \\ b_0 + b_2 + b_3 &= u_1 \\ b_1 + b_3 + b_4 &= u_2 \\ b_0 + b_2 + b_4 &= u_3 \\ b_1 + b_2 + b_3 &= u_4 \end{aligned}$$

Solving the above equations, we obtain

$$\begin{aligned}b_0 &= u_2 + u_3 + u_4 \\b_1 &= u_1 + u_2 + u_3 \\b_2 &= u_0 + u_1 + u_2 \\b_3 &= u_0 + u_3 + u_4 \\b_4 &= u_0 + u_1 + u_4\end{aligned}$$

Since $b = uH_1^{-1}$, the above equations can be write in matrix form ads

$$b = [u] \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Thus,

$$H_1^{-1} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$H_2 H_1^{-1} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

The generator matrix $G = [H_2 H_1^{-1} \ I_k]$

$$G = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The codeword can be generated as $C = mG$.

$$C = [1 \ 0 \ 0 \ 0 \ 1] \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$C = [1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1]$$

$$CH^T = [1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1] \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} = [0 \ 0 \ 0 \ 0 \ 0]$$

8.5 Efficient Encoding of LDPC Codes

The preprocessing method discussed in Sect. 8.4.1 for finding a generator matrix G for a given H can be used for encoding any arbitrary message bits vector of size $1 \times m$. However, it has a complexity of $O(n^2)$ [9]. LDPC code can be encoded using the parity-check matrix directly by using the efficient encoding method [6] which has a complexity of $O(n)$. The stepwise procedure of efficient coding of LDPC coding [10] is as follows:

Step 1: By performing row and column permutations, the nonsingular parity check matrix H is to be brought into a lower triangular form indicated in Fig. 8.5.

More precisely, the H matrix is brought into the form

$$H_t = \begin{bmatrix} A & B & T \\ C & D & E \end{bmatrix} \quad (8.17)$$

with a gap length g as small as possible.

Where A is $(m - g) \times (n - m)$ matrix, B is $(m - g) \times g$ matrix, T is $(m - g) \times (m - g)$ matrix, C is $g \times (n - m)$ matrix, D is $g \times g$ matrix and E is $g \times (m - g)$ matrix. All of these matrices are sparse and T is lower triangular with ones along the diagonal.

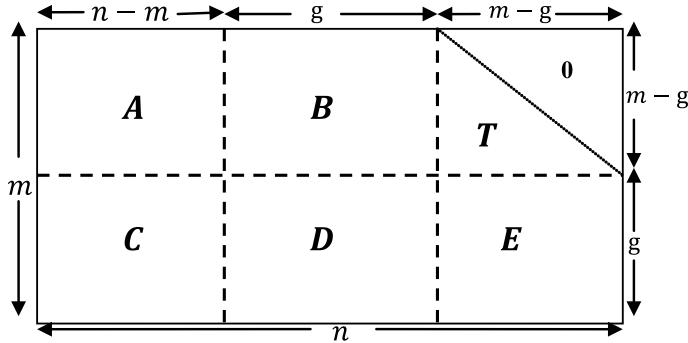


Fig. 8.5 The parity-check matrix in approximate lower triangular form

Step 2: Premultiply H_t by $\begin{bmatrix} I_{m-g} & 0 \\ -ET^{-1} & I_g \end{bmatrix}$

$$\begin{bmatrix} I_{m-g} & 0 \\ -ET^{-1} & I_g \end{bmatrix} \begin{bmatrix} A & B & T \\ C & D & E \end{bmatrix} = \begin{bmatrix} A & B & T \\ -ET^{-1}A + C & -ET^{-1}B + D & 0 \end{bmatrix} \quad (8.18)$$

In order to check that $-ET^{-1}B + D$ is nonsingular. It is to be ensured by performing column permutations further.

Step 3: Obtain p_1 using the following

$$p_1^T = -\emptyset^{-1}(-ET^{-1}A + C)s^T \quad (8.19)$$

where

$\emptyset = -ET^{-1}B + D$ and s is message vector.

Step 4: Obtain p_2 using the following

$$p_2^T = -T^{-1}(As^T + Bp_1^T) \quad (8.20)$$

Step 5: Form the code vector c as

$$c = [s \ p_1 \ p_2] \quad (8.21)$$

p_1 holds the first g parity and p_2 contains the remaining parity bits.

Example 8.6 Construct LDPC code word for the following parity check matrix with the message vector $m = [1 \ 0 \ 0 \ 0 \ 1]$.

$$H = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Solution

Step 1: Second and third rows and third and tenth columns are swapped to obtain

$$H_t = \left[\begin{array}{cc|cc|cc|ccc} 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ \hline 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \end{array} \right]$$

Step 2:

$$T^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad E = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} I_{m-g} & 0 \\ -ET^{-1} & I_g \end{bmatrix} \left[\begin{array}{cc|cc|cc|ccc} 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ \hline 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \end{array} \right] = \left[\begin{array}{cc|cc|cc|ccc} 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ \hline 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{array} \right]$$

Step 3:

$$p_1^T = -[-ET^{-1}B + D]^{-1}(-ET^{-1}A + C)s^T = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Step 4:

$$p_2^T = -T^{-1}(As^T + Bp_1^T) = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

Step 5:

$$c = [s \ p_1 \ p_2] = [1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 1]$$

8.5.1 Efficient Encoding of LDPC Codes Using MATLAB

The following example illustrates the efficient encoding of LDPC codes using MATLAB.

Example 8.7 Write a MATLAB program to encode a random message vector with the following parity check matrix

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Program 8.4 MATLAB program for efficient encoding of LDPC Codes

```
%Program for efficient encoding of LDPC Codes
clear all;clc;
H=[1 1 0 1 1 0 0 1 0 0;0 1 1 0 1 1 1 0 0 0;0 0 0 1 0 0 0 1 1 1;1 1 0 0 0 1 1 0 0 1
1 0 ;0 0 1 0 0 1 0 1 0 1];
Hlt=[1 1 0 1 1 0 0 1 0 0;0 0 0 1 0 1 0 1 1 0;0 1 1 0 1 0 1 0 0 1;1 1 0 0 0 0 0
1 0 1 1 ;0 0 1 0 0 1 0 1 0 1];
msg = round(rand(1,size(H,1)));
p=ldpcclinearencode(Hlt,msg);
c=[msg p ];
% Checking c*Hlt'=0;
cs = mod(c'*Hlt',2);
if sum(cs)~= 0
    disp('Error')
```

MATLAB function program `ldpcclinearencode` used in Program 8.4.

```
function p=ldpcclinearencode(H,msg);
n = size(H,2);
m = size(H,2) - size(H,1);
Hr = H(:,end);
% Find the 'gap' length
for i=1:size(H,2)
    if Hr(i) == 1
        g = i;
        break;
    end
end
```

```

g = size(H,1) - g;
% Extracting the submatrices A, B, C, D, E and T
A = H(1:m-g,1:n-m);
B = H(1:m-g,n-m+1:n-m+g);
T = H(1:m-g,n-m+g+1:end);
C = H(m-g+1:end,1:n-m);
D = H(m-g+1:end,n-m+1:n-m+g);
E = H(m-g+1:end,n-m+g+1:end);
% Calculate p1 and p2
invT = (inv(T)); % or abs(inv(T)) ?
ET1 = -(E*invT);
phi = ET1*B + D;
xtra = ET1*A + C;
p1 = mod(phi*xtra*(msg'),2)';
p2 = mod(invT*(A*(msg') + B*(p1')),2)';
p = [p1 p2];

```

8.6 LDPC Decoding

In the LDPC decoding, the notation B_j is used to represent the set of bits in the parity check equation of H , and the notation A_i is used to represent the parity check equations for the i th bit of the code. Consider the following parity check matrix

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \quad (8.22)$$

For the above parity check matrix, we get

$$\begin{aligned} B_1 &= \{1, 2, 3\}, & B_2 &= \{1, 4, 5\}, & B_3 &= \{2, 4, 6\}, & B_4 &= \{3, 5, 6\}, \\ A_1 &= \{1, 2\}, & A_2 &= \{1, 3\}, & A_3 &= \{1, 4\}, & A_4 &= \{2, 3\}, & A_5 &= \{2, 4\}, & A_6 &= \{3, 4\} \end{aligned}$$

8.6.1 LDPC Decoding on Binary Erasure Channel Using Message Passing Algorithm

The message passing algorithms are iterative decoding algorithms which passes the messages back and forward between the bit and check nodes iteratively until the process is stopped. The message labeled M_i indicates 0 or 1 for known bit values and e for erased bit the stepwise procedure for LDPC decoding on BEC is as follows:

- Step 1. Set $M = y$, find B_j and A_i of H
 Step 2. $iter = 1$
 Step 3. If all messages into check j other than M_i are known, compute all check sums by using the following expression

$$E_{j,i} = \sum_{i' \in B_j, i' \neq i} (M_{i'} \mod 2)$$

- else $E_{j,i} = e$
 Step 4. If $M_i = e$ and if $j \in A_i$ subject to $E_{j,i} \neq e$, set $M_i = E_{j,i}$.
 Step 5. If all M_i are known or $iter = iter_{max}$, stop, else
 Step 6. $iter = iter + 1$, go to Step 3.

Example 8.8 For the parity check matrix given by Eq. (8.22), $c = [1\ 0\ 1\ 1\ 0\ 1]$ is a valid code word since $cH^T = 0$. If the code word is sent through BEC, the received vector is $y = [1\ 0\ e\ e\ e\ 1]$. Decode the received vector to recover the erased bits using message passing algorithm.

Solution For Step 3 of the algorithm, the first check node is joined to the first, second and third bit nodes having incoming messages 1, 0 and e . This check node has one incoming e message from the third bit node. Hence, we can calculate the outgoing message $E_{1,3}$ on the edge from the first check node to the third bit node:

$$\begin{aligned} E_{1,3} &= M_1 + M_2 \\ &= 1 \oplus 0 \\ &= 1. \end{aligned}$$

The second check node is joined to the first, fourth and fifth bit nodes having incoming messages 1, e and e . As this check node has two e messages, the outgoing messages from this check node are all e .

The third check node is joined to the second, fourth and sixth bits receiving incoming messages 0, e and 1. This check node has one incoming e message from the fourth bit node. Hence, the outgoing message $E_{1,3}$ on the edge from the third check node to the fourth bit node is given by

$$\begin{aligned} E_{3,4} &= M_2 + M_6 \\ &= 0 \oplus 1 \\ &= 1. \end{aligned}$$

The fourth check node includes the third, fifth and sixth bits and receives e , e and 1 messages. Since this check node receives two e messages, the outgoing messages from this check node are all e .

In Step 4 of the algorithm, each bit node with an unknown value updates its value uses its incoming messages. The third bit is unknown and has incoming messages

1 ($E_{1,3}$) and e ($E_{4,3}$) and hence the third bit value becomes 1. The fourth bit is not known and it is set to 1 as it has incoming messages 1 ($E_{2,4}$) and e ($E_{3,4}$). The fifth bit is also unknown but its value cannot be changed because has e ($E_{2,5}$) and e ($E_{4,5}$) as incoming messages. Thus, at the end of the Step 4

$$M = [1 \ 0 \ 1 \ 1 \ e \ 1].$$

Since the fifth bit is remaining unknown and hence the algorithm is to be continued. In the second iteration, in the Step 3 of the algorithm, the second check node is joined to the first, fourth and fifth bit nodes and so this check node has one incoming e message, M_5 . Hence, the outgoing message from this check node becomes

$$\begin{aligned} E_{2,5} &= M_1 + M_4 \\ &= 1 \oplus 1 \\ &= 0. \end{aligned}$$

The fourth check node is joined to the third, fifth and sixth bit nodes having one incoming e message, M_5 . The outgoing message from this check to the sixth bit node, $E_{4,6}$, is the value of the sixth codeword bit:

$$\begin{aligned} E_{4,5} &= M_3 + M_6 \\ &= 1 \oplus 1 \\ &= 0. \end{aligned}$$

In the second iteration, in the Step 4, the unknown fifth bit is changed to 0 as it has $E_{2,5}$ and $E_{4,5}$ as incoming messages with value 0. The algorithm is stopped and the decoded code word is

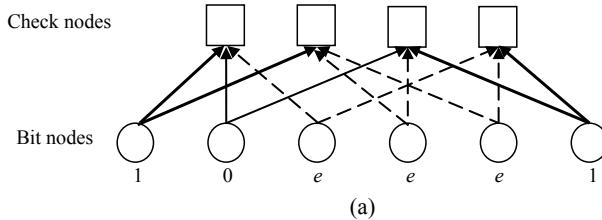
$$\hat{c} = M = [1 \ 0 \ 1 \ 1 \ 0 \ 1]$$

as the decoded codeword Fig. 8.6 shows the graphical representation of message passing decoding.

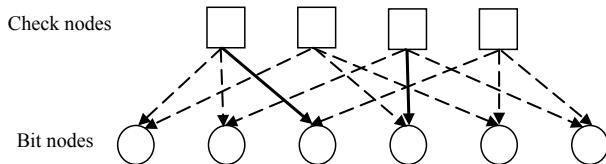
8.6.2 LDPC Decoding on Binary Erasure Channel Using MATLAB

The following example illustrates decoding of LDPC codes on BEC using MATLAB.

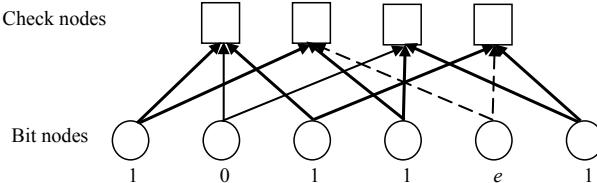
Example 8.9 Write a MATLAB program to implement LDPC decoding on BEC by assuming received vector $y = [1 \ e \ e \ e \ e \ 1 \ 0 \ 0 \ 0 \ 1 \ e \ 1 \ e \ e \ e \ 1]$ when the following parity is used to encode the codeword.

Initialization:

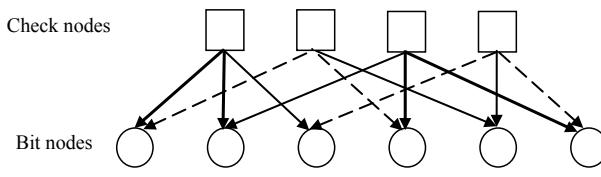
(a)

First iteration:

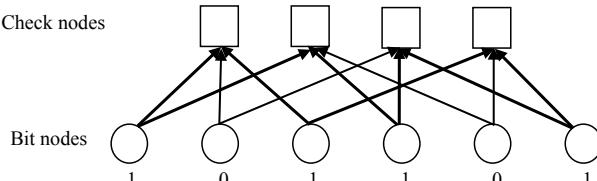
(b) Check messages



(c) Bit messages

Second iteration:

(d) Check messages



(e) Bit messages

Fig. 8.6 Decoding of received vector $y = [1 \ 0 \ e \ e \ e \ 1]$ using message passing. The dark line corresponds to message bit 1, solid line corresponds to message bit 0, and the broken line corresponds to erasure bit e

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Solution The following MATLAB program decodes the received vector y . In this program, known bit values are indicated by 1 or 0 and erased bit is indicated by -1 .

Program 8.5 MATLAB program Decoding of LDPC Codes on BEC

```
% Program for Decoding of LDPC Codes on BEC %%%
clear all;clc;
% y=[0 1 -1 -1 -1];
y=[1 -1 -1 -1 -1 1 0 0 0 1 -1 1 -1 -1 1];
M=y;
% H=[1 1 0 1 0 0;0 1 1 0 1 0;1 0 0 0 1 1 ; 0 0 1 1 0 1];
H=[1 1 1 0 1 1 0 0 0 0 0 0 0 0 0;1 0 0 1 1 1 0 1 0 0 0 0 0 0 0;
    0 0 1 1 1 0 0 0 1 0 0 0 0 0 0;0 1 1 1 1 0 0 0 0 1 0 0 0 0 0;
    0 1 0 0 0 0 0 0 1 1 0 0 0 0 0;1 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0;
    1 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0;1 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0;
    1 1 1 0 1 0 0 0 0 0 0 0 0 1 0 0;0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0;
    1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1];
[N1 N2]=size(H);
iter=input('enter the number of iterations')
for i=1:iter
    for j=1:N1
        ci = find(H(j, :));
        d=find(M(ci)~-=-1);
        d1=find(M(ci)==-1);
        if ((length(d)>=2) & (length(d1)==1))
            E(j,ci(d1))=mod(sum(M(ci(d))),2);
        else
            E(j,ci(d1))=-1;
        end
    end
end
```

```

for j=1:N2
    ri = find(H(:,j));
    if(M(j)==-1)
        for ii=1:length(ri)
            if( E(ri(ii),j)~=-1)
                M(j)=E(ri(ii),j);
            end
        end
    end
end
end

```

8.6.3 Bit-Flipping Decoding Algorithm

The received symbols are hard decoded into 1s and 0s to form a binary received vector y . In each iteration, it computes all check sums, as well as the number of unsatisfied parity checks involving each of the n bits of the vector y . Next, the bits of y are flipped if they involve in the largest number of unsatisfied parity checks. The process is to be repeated until all check sums are satisfied or reaches a predetermined number of iterations. The stepwise procedure of the Bit-flipping decoding algorithm is as follows:

- Step 1. Set $M = y$, define B_j to represent the j th parity check equation of H
- Step 2. $l = 0$
- Step 3. Compute all check sums by using the following expression

$$E_{j,i} = \sum_{i' \in B_j, i' \neq i} (M_{i'} \mod 2) \quad (8.23)$$

- Step 4. Compute the number of unsatisfied parity checks involving each of n bits of message
- Step 5. Flip the bits of message when they are involved in largest number of unsatisfied parity checks. The flipping on i th bit can be performed by using

$$M_i = (y_i + 1 \mod 2) \quad (8.24)$$

- Step 6. Compute s as follows

$$s = (MH^T) \mod 2 \quad (8.25)$$

- Step 7. If $s = 0$ or $l = l_{\max}$, stop, else
- Step 8. $l = l + 1$, go to Step 3.

Example 8.10 For the parity check matrix given by Eq. (8.22), $c = [1 \ 0 \ 1 \ 1 \ 0 \ 1]$ is a valid code word since $cH^T = 0$. If the code word is sent through AWGN channel,

the received vector after a detector hard decision is $y = [0\ 0\ 1\ 1\ 0\ 1]$. Decode the received vector using bit flipping algorithm.

Solution The decoder makes a hard decision on each codeword bit and returns

$$y = [0\ 0\ 1\ 1\ 0\ 1]$$

Step 1: Initializing $M_i = y_i$, so

$$M = [0\ 0\ 1\ 1\ 0\ 1].$$

Step 2: $l = 0$

Step 3: The check messages are calculated. The first check node is joined to the first, second and third bit nodes $B_1 = \{1, 2, 3\}$ and so that the messages from the first check node are

$$\begin{aligned} E_{1,1} &= M_2 \oplus M_3 = 0 \oplus 1 = 1, \\ E_{1,2} &= M_1 \oplus M_3 = 0 \oplus 1 = 1, \\ E_{1,3} &= M_1 \oplus M_2 = 0 \oplus 0 = 0, \end{aligned}$$

The second check includes the first, fourth and fifth bits, $B_2 = \{1, 4, 5\}$ and so the messages from the second check are

$$\begin{aligned} E_{2,1} &= M_4 \oplus M_5 = 1 \oplus 0 = 1, \\ E_{2,4} &= M_1 \oplus M_5 = 0 \oplus 0 = 0, \\ E_{2,5} &= M_1 \oplus M_4 = 0 \oplus 1 = 1. \end{aligned}$$

The third check includes the second, fourth and sixth bits, $B_3 = \{2, 4, 6\}$, and so the messages from the second check are

$$\begin{aligned} E_{3,2} &= M_4 \oplus M_6 = 1 \oplus 1 = 0, \\ E_{3,4} &= M_2 \oplus M_6 = 0 \oplus 1 = 1, \\ E_{3,6} &= M_2 \oplus M_4 = 0 \oplus 1 = 1, \end{aligned}$$

The fourth check includes the third, fifth and sixth bits, $B_4 = \{3, 5, 6\}$, and so the messages from the second check are

$$\begin{aligned} E_{4,3} &= M_5 \oplus M_6 = 0 \oplus 1 = 1, \\ E_{4,5} &= M_3 \oplus M_6 = 1 \oplus 1 = 0. \\ E_{4,6} &= M_3 \oplus M_5 = 1 \oplus 0 = 1. \end{aligned}$$

Step 4: The first bit has messages 1 and 1 from the first and second checks respectively and 0 from the channel. Thus the majority of the messages into the first bit node

indicate a value different from the received value. The second bit has messages 1 and 0 from the first and third checks respectively and 0 from the channel, so it retains its received value. The third bit has messages 0 and 1 from the first and fourth checks respectively and 1 from the channel, so it retains its received value. The fourth bit has messages 0 and 1 from the second and third checks respectively and 1 from the channel, so it retains its received value. The fifth bit has messages 1 and 0 from the second and fourth checks respectively and 0 from the channel, so it retains its received value. The sixth bit has messages 1 and 1 from the third and fourth checks respectively and 1 from the channel, so it retains its received value. Thus the majority of the messages into the first bit node indicate a value different from the received value

Step 5: Hence the first bit node flips its value. The new bit node to check node messages are thus given by

$$M = [1 \ 0 \ 1 \ 1 \ 0 \ 1].$$

Step 6: Compute $s = (MH^T) \bmod 2$

$$s = \left([1 \ 0 \ 1 \ 1 \ 0 \ 1] \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \right) \bmod 2 = [0 \ 0 \ 0 \ 0]$$

there are thus no unsatisfied parity-check equations and so the algorithm halts and returns

$$\hat{c} = M = [1 \ 0 \ 1 \ 1 \ 0 \ 1]$$

as the decoded codeword. The received vector has therefore been correctly decoded without requiring an explicit search over all possible codewords. Hence the process is stopped.

8.6.4 Bit-Flipping Decoding Using MATLAB

The following example illustrates the bit flipping decoding of LDPC codes using MATLAB.

Example 8.11 Write a MATLAB program to implement bit flipping decoding by assuming received vector $y = [0\ 1\ 1\ 0\ 1\ 1]$ when the following parity check matrix

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

is used to encode the codeword.

Solution The following MATLAB program decodes the received vector y .

Program 8.6 MATLAB program for Bit Flipping Decoding of LDPC Codes

```
% Program for Bit Flipping Decoding of LDPC Codes
clear all;clc;
y=[0 1 1 0 1 1];
H=[1 1 0 1 0 0;0 1 1 0 1 0;1 0 0 0 1 1 ; 0 0 1 1 0 1];
[N1 N2]=size(H);
iter=input('enter the number of iterations')
for i=1:iter
for j=1:N1
    ci = find(H(j, :));
    for k=1:length( ci)
        E(j, ci(k)) = mod(sum(y( ci )) + y( ci (k)), 2);
    end
end
for j=1:N2
    ri = find(H(:, j));
    numberofones=length(find(E(ri,j)));
    numberofzeros=length(ri)-numberofones;
    if(numberofones==numberofzeros)
        yd(j)=y(j);
    elseif(numberofones > numberofzeros)
        yd(j)=1;
    elseif(numberofones < numberofzeros)
        yd(j)=0;
    end
end
y=yd;
end
```

The output of the above program gives the decoded vector

$$\hat{c} = y_d = [0\ 0\ 1\ 0\ 1\ 1]$$

Example 8.12 A valid code word is $c = [0\ 0\ 1\ 0\ 0\ 1]$ for the following parity check matrix

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

If the code word is transmitted over AWGN channel, the received vector after detector hard decision is $y = [1\ 0\ 1\ 0\ 0\ 1]$. Decode the received vector by bit-flipping using MATLAB and comment on the result.

Solution The Program 8.6 is run with the H matrix and the received vector. The output of the program gives the decoded vector $\hat{c} = y_d = [0\ 1\ 1\ 0\ 0\ 1]$. The received vector is not decoded correctly due to the girth 4 in the H matrix.

8.7 Sum Product Decoding

The sum product algorithm is similar to the bit-flipping algorithm as described in the previous section, but the messages representing each decision (whether the bit value is 1 or 0) are now probabilities. Bit-flipping decoding accepts an initial hard decision on the received bits as input and the sum-product algorithm is a soft decision message passing algorithm which accepts the probability of each received bit as input. The input channel or received bit probabilities are known in advance before the LDPC decoder was operated and so they are also called as the a priori probabilities of the received bit. In the sum product decoder, the extrinsic information passed between nodes is also probabilities. The extrinsic information between check node j and bit node i is denoted by $E_{j,i}$. The $E_{j,i}$ gives the probability for the bit c_i to be 1 that causes the parity check equation j is satisfied. The $E_{j,i}$ can not be defined if the bit i not included in j as there will be no extrinsic information between check node j and bit node i .

The probability that an odd number of the bits in that parity check equation are 1s is given by

$$P_{j,i}^{\text{ext}} = \frac{1}{2} - \frac{1}{2} \prod_{i' \in B_j, i' \neq i} (1 - 2P_{j,i'}) \quad (8.26)$$

Which is the probability that a parity check equation is satisfied for the bit c_i to be 1. The probability that the parity check equation is satisfied for the bit c_i to be 0 becomes $1 - P_{j,i}^{\text{ext}}$.

The metrics for a binary variable is represented by the following log likelihood ratio (LLR)

$$L(x) = \log \frac{p(x=0)}{p(x=1)} \quad (8.27)$$

where by log we mean \log_e . The sign of $L(x)$ provides a hard decision on x and magnitude $|L(x)|$ is the reliability of this decision. Translating from log likelihood ratios back to probabilities,

$$p(x=1) = \frac{e^{-L(x)}}{1 + e^{-L(x)}} \quad (8.28)$$

$$p(x=0) = \frac{e^{L(x)}}{1 + e^{-L(x)}} \quad (8.29)$$

when probabilities need to be multiplied, log likelihood ratios need only be added and by this the complexity of the sum product decoder is reduced. This makes the benefits of the logarithmic representation of probabilities. The extrinsic information from check node j to bit node i is expressed as a log likelihood ratio,

$$E_{j,i} = L(P_{j,i}^{\text{ext}}) = \log \frac{1 - P_{j,i}^{\text{ext}}}{P_{j,i}^{\text{ext}}} \quad (8.30)$$

Now

$$\begin{aligned} E_{j,i} &= \log \frac{\frac{1}{2} + \frac{1}{2} \prod_{i' \in B_j, i' \neq i} (1 - 2P_{j,i'})}{\frac{1}{2} - \frac{1}{2} \prod_{i' \in B_j, i' \neq i} (1 - 2P_{j,i'})} \\ &= \log \frac{1 + \prod_{i' \in B_j, i' \neq i} \left(1 - 2 \frac{e^{-M_{j,i'}}}{1 + e^{-M_{j,i'}}}\right)}{1 - \prod_{i' \in B_j, i' \neq i} \left(1 - 2 \frac{e^{-M_{j,i'}}}{1 + e^{-M_{j,i'}}}\right)} \\ &= \log \frac{1 + \prod_{i' \in B_j, i' \neq i} \left(\frac{1 - e^{-M_{j,i'}}}{1 + e^{-M_{j,i'}}}\right)}{1 - \prod_{i' \in B_j, i' \neq i} \left(\frac{1 - e^{-M_{j,i'}}}{1 + e^{-M_{j,i'}}}\right)} \end{aligned} \quad (8.31)$$

where $M_{j,i'} \triangleq L(P_{j,i'}) = \log \frac{1 - P_{j,i'}}{P_{j,i'}}$.

Using the relationship

$$\tanh \frac{1}{2} \log \left(\frac{1 - p}{p} \right) = 1 - 2p \quad (8.32)$$

gives

$$E_{j,i} = \log \frac{1 + \prod_{i' \in B_j, i' \neq i} \tanh(M_{j,i'}/2)}{1 - \prod_{i' \in B_j, i' \neq i} \tanh(M_{j,i'}/2)} \quad (8.33)$$

Alternatively, using the relationship

$$2 \tanh^{-1} p = \log \frac{1 + p}{1 - p} \quad (8.34)$$

Then

$$E_{j,i} = 2 \tanh^{-1} \prod_{i' \in B_j, i' \neq i} \tanh(M_{j,i'}/2) \quad (8.35)$$

The above equation is numerically challenging due to the presence of the product of the \tanh and \tanh^{-1} functions. Following Gallager, we can improve the situation as follows. First, factor M_{ji} into its sign and magnitude (or bit value and bit reliability);

$$M_{ji} = \alpha_{ji} \beta_{ji} \quad (8.36)$$

$$\alpha_{ji} = \text{sign}(M_{ji}) \quad (8.36a)$$

$$\beta_{ji} = |M_{ji}| \quad (8.36b)$$

So that Eq. (8.35) may be rewritten as

$$\tanh\left(\frac{1}{2}M_{ji}\right) = \prod_{i'} \alpha_{ji'} \cdot \prod_{i' \in B_j, i' \neq i} \tanh\left(\frac{1}{2}\beta_{ji'}\right) \quad (8.37)$$

We then have

$$\begin{aligned} E_{ji} &= \prod_{i'} \alpha_{ji'} \cdot 2 \tanh^{-1} \left(\prod_{i'} \tanh\left(\frac{1}{2}\beta_{ji'}\right) \right) \\ &= \prod_{i'} \alpha_{ji'} \cdot 2 \tanh^{-1} \log^{-1} \log \left(\prod_{i'} \tanh\left(\frac{1}{2}\beta_{ji'}\right) \right) \\ &= \prod_{i'} \alpha_{ji'} \cdot 2 \tanh^{-1} \log^{-1} \sum_{i'} \log \left(\tanh\left(\frac{1}{2}\beta_{ji'}\right) \right) \end{aligned} \quad (8.38)$$

This yields a new form for Eq. (8.38) as

$$E_{ji} = \prod_{i'} \alpha_{ji'} \cdot \phi \left(\sum_{i'} \phi(\beta_{ji'}) \right) \quad (8.39)$$

where $\phi(x)$ is defined as

$$\phi(x) = -\log[\tanh(x/2)] = \log\left(\frac{e^x + 1}{e^x - 1}\right) \quad (8.40)$$

Using the fact that $\phi^{-1}(x) = \phi(x)$ when $x > 0$.

Each bit node has access to the input LLR, L_i , and to the LLRs from every connected check node. The total LLR of the i th bit is the sum of these LLRs:

$$L_i^{\text{total}} = L_i + \sum_{j \in A_i} E_{ji} \quad (8.41)$$

The hard decision on the received bits is simply given by the signs of the L_i^{total} . Check whether the parity check equations are satisfied (thus $\hat{c}H^T = 0$ is also a stopping criterion for sum product decoding), if not satisfied update M_{ji}

$$M_{ji} = \sum_{j' \in A_i, j' \neq j} E_{j'i} + L_i \quad (8.42)$$

The algorithm outputs the estimated a posteriori bit probabilities of the received bits as log likelihood ratios.

The sum-product decoder immediately stops whenever a valid codeword has been found by a checking of whether the parity check equations are satisfied (i.e., $\hat{c}H^T = 0$) or allowed maximum number of iterations achieved. The decoder is initialized by setting all VN messages M_{ji} equal to

$$L_i = L(c_i | y_i) = \log\left(\frac{\Pr(c_i = 0 | y_i)}{\Pr(c_i = 1 | y_i)}\right) \quad (8.43)$$

For all j, i for which $h_{ij} = 1$. Here, y_j represents the channel value that was actually received, that is, it is not a variable here. The L_i for different channels can be computed as [10]

BEC

In this case, $y_j \in \{0, 1, e\}$

$$L_i = L(c_i | y_i) = \begin{cases} +\infty & y_j = 0, \\ -\infty & y_j = 1, \\ 0 & y_j = e. \end{cases} \quad (8.44)$$

BSC

In this case, $y_j \in \{0, 1\}$, we have

$$L_i = L(c_i | y_i) = (-1)^{y_j} \log\left(\frac{1 - P}{P}\right) \quad (8.45)$$

The knowledge of crossover probability P is necessary.

BI-AWGNC

The i th received sample is $y_i = x_i + n_i$, where the n_i are independent and normally distributed as $\mathcal{N}(0, \sigma^2)$. $\sigma^2 = \frac{N_0}{2}$, where N_0 is the noise density.

Then we can easily show that

$$\Pr(x_i = x | y_i) = \frac{1}{1 + \exp(-4y_i x / N_0)} \quad (8.46)$$

where $x \in \{\pm 1\}$ and, from this, that

$$L(c_i | y_i) = 4y_i / N_0 \quad (8.47)$$

An estimate of N_0 is necessary in practice.

Rayleigh

The model for Rayleigh fading channel is similar to that of the AWGNC: $y_i = \alpha_i x_i + n_i$, where $\{\alpha_i\}$ are independent Rayleigh random variable with unity variance. The channel transition probability can be expressed by

$$P(x_i = x | y_i) = \frac{1}{1 + \exp(-4\alpha_i y_i x / N_0)}$$

Then

$$L(c_i | y_i) = 4\alpha_i y_i / N_0 \quad (8.48)$$

The estimates of α_i and σ^2 are necessary in practice.

Now, the stepwise procedure for the log domain sum product algorithm is given in the following Sect. 8.8.

8.7.1 Log Domain Sum-Product Algorithm (SPA)

- Step 1. Initialization: for all i , initialize L_i according to Eq. (8.44) for the appropriate channel model. Then, for all i, j for which $h_{i,j} = 1$ set $M_{ji} = L_i$, and $l = 0$. Define B_j to represent the set of bits in the j th parity check equation of H and A_i to represent the parity check equations for the i th bit of the code.
- Step 2. CN update: compute out going CN message E_{ji} for each CN using Eqs. (8.36), (8.39) and (8.40).

$$\begin{aligned} M_{ji} &= \alpha_{ji} \beta_{ji} \\ \alpha_{ji} &= \text{sign}(M_{ji}), \end{aligned}$$

$$\begin{aligned}\beta_{ji} &= |M_{ji}| \\ E_{ji} &= \prod_{i'} \alpha_{ji'} \cdot \phi\left(\sum_{i'} \phi(\beta_{ji'})\right) \\ \phi(x) &= -\log[\tanh(x/2)] = \log\left(\frac{e^x + 1}{e^x - 1}\right)\end{aligned}$$

Step 3. LLR total: For $i = 0, 1, \dots, N - 1$ compute total LLR using Eq. (8.41)

$$L_i^{\text{total}} = L_i + \sum_{j \in A_i} E_{ji}$$

Step 4. Stopping criteria: For $i = 0, 1, \dots, N - 1$, set

$$\hat{c}_i = \begin{cases} 1 & \text{if } L_i^{\text{total}} < 0, \\ 0 & \text{else,} \end{cases}$$

To obtain \hat{c} . If $\hat{c}H^T = 0$ or the number of iterations equals the maximum limit ($l = l_{\max}$) stop;
else

Step 5. VN update: compute outgoing VN message M_{ji} for each VN using Eq. (8.42)

$$M_{ji} = L_i + \sum_{j' \in A_i, j' \neq j} E_{j'i} \cdot l = l + 1 \text{ go to Step 2.}$$

8.7.2 The Min-Sum Algorithm

Consider Eq. (8.39) for E_{ji} . It can be noted from the shape of $\phi(x)$ that the largest term in the sum corresponds to the smallest β_{ji} . Hence, assuming that this term dominates the sum, the following relation is obtained [10]

$$\phi\left(\sum_{i'} \phi(\beta_{ji'})\right) \simeq \phi\left(\phi\left(\min_{i'} \beta_{ji'}\right)\right) = \min_{i'} \beta_{ji'} \quad (8.49)$$

Thus, the Min-Sum algorithm is simply the log domain SPA with Step 2 replaced by

$$\begin{aligned}M_{ji} &= \alpha_{ji} \beta_{ji} \\ \alpha_{ji} &= \text{sign}(M_{ji}), \\ \beta_{ji} &= |M_{ji}|\end{aligned}$$

$$E_{ji} = \prod_{i'} \alpha_{ji'} \cdot \min_{i'} \beta_{ji'}$$

It can also be shown that, in the AWGNC case, the initialization $M_{ji} = 4y_i/N_0$ may be replaced by $M_{ji} = y_i$ when the simplified log domain sum product algorithm is employed. The advantage, of course, is that an estimate of the noise power N_0 is unnecessary in this case.

Example 8.13 A code word generated using the parity check matrix $H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$ is sent through AWGN channel with $N_o = 0.3981$, the received vector is $\mathbf{y} = [-0.9865 \ 0.3666 \ 0.4024 \ 0.7638 \ 0.2518 \ -1.6662]$. Decode the received vector using the sum-product algorithm.

Solution

$$\begin{aligned} L &= -4 \frac{\mathbf{y}}{N_0} \\ &= [9.9115 \ -3.6830 \ -4.0430 \ -7.6738 \ -2.5295 \ 16.7415] \end{aligned}$$

To begin decoding we set

$$M_{j,i} = L_i$$

The first bit is included in the first and second checks and so $M_{1,1}$ and $M_{2,1}$ are initialized to L_1 :

$$M_{1,1} = L_1 = 9.9115 \quad \text{and} \quad M_{2,1} = L_1 = 9.9115.$$

Repeating this for the remaining bits gives:

- For $i = 2$, $M_{1,2} = L_2 = -3.6830$, $M_{3,2} = L_2 = -3.6830$;
- For $i = 3$, $M_{1,3} = L_3 = -4.0430$, $M_{4,3} = L_3 = -4.0430$;
- For $i = 4$, $M_{2,4} = L_4 = -7.6738$, $M_{3,4} = L_4 = -7.6738$;
- For $i = 5$, $M_{2,5} = L_5 = -2.5295$, $M_{4,5} = L_5 = -2.5295$;
- For $i = 6$, $M_{3,6} = L_6 = 16.7415$, $M_{4,6} = L_6 = 16.7415$;

Now the extrinsic probabilities are calculated for the check to bit messages, the first parity check includes the first, second and third bits and so the extrinsic probability from the first check node to the first bit node depends on the probabilities of the second and third bits:

$$\begin{aligned} E_{1,1} &= \log \frac{1 + \tanh(M_{1,2}/2) \tanh(M_{1,3}/2)}{1 - \tanh(M_{1,2}/2) \tanh(M_{1,3}/2)} \\ &= \log \frac{1 + \tanh(-3.6830/2) \tanh(-4.0430/2)}{1 - \tanh(-3.6830/2) \tanh(-4.0430/2)} = 3.1542 \end{aligned}$$

Similarly, the extrinsic probability from the first check node to the second bit node depends on the probabilities of the first and third bits:

$$\begin{aligned} E_{1,2} &= \log \frac{1 + \tanh(M_{1,1}/2) \tanh(M_{1,3}/2)}{1 - \tanh(M_{1,1}/2) \tanh(M_{1,3}/2)} \\ &= \log \frac{1 + \tanh(9.9115/2) \tanh(-4.0430/2)}{1 - \tanh(9.9115/2) \tanh(M_{1,3}/2)} = -4.0402 \end{aligned}$$

And the extrinsic probability from the first check node to the third bit node depends on the LLRs sent from the first and second bit nodes to the first check node:

$$\begin{aligned} E_{1,3} &= \log \frac{1 + \tanh(M_{1,1}/2) \tanh(M_{1,2}/2)}{1 - \tanh(M_{1,1}/2) \tanh(M_{1,2}/2)} \\ &= \log \frac{1 + \tanh(9.9115/2) \tanh(-3.6830/2)}{1 - \tanh(9.9115/2) \tanh(-3.6830/2)} = -3.681 \end{aligned}$$

Next, the second check node connects to the first, fourth and fifth bit nodes and so the extrinsic LLRs are

$$\begin{aligned} E_{2,1} &= \log \frac{1 + \tanh(M_{2,4}/2) \tanh(M_{2,5}/2)}{1 - \tanh(M_{2,4}/2) \tanh(M_{2,5}/2)} \\ &= \log \frac{1 + \tanh(-7.6738/2) \tanh(-2.5295/2)}{1 - \tanh(-7.6738/2) \tanh(-2.5295/2)} = 2.5237 \\ E_{2,4} &= \log \frac{1 + \tanh(M_{2,1}/2) \tanh(M_{2,5}/2)}{1 - \tanh(M_{2,1}/2) \tanh(M_{2,5}/2)} \\ &= \log \frac{1 + \tanh(9.9115/2) \tanh(-2.5295/2)}{1 - \tanh(9.9115/2) \tanh(-2.5295/2)} = -2.5289 \\ E_{2,5} &= \log \frac{1 + \tanh(M_{2,1}/2) \tanh(M_{2,4}/2)}{1 - \tanh(M_{2,1}/2) \tanh(M_{2,4}/2)} \\ &= \log \frac{1 + \tanh(9.9115/2) \tanh(-7.6738/2)}{1 - \tanh(9.9115/2) \tanh(-7.6738/2)} = -7.5724 \end{aligned}$$

Similarly for the remaining check nodes

$$\begin{aligned} E_{3,2} &= -7.6737, \quad E_{3,4} = -3.6830, \quad E_{3,6} = 3.6647, \\ E_{4,3} &= -2.5295, \quad E_{4,5} = -4.0430, \quad E_{4,6} = -4.0430. \end{aligned}$$

To check for a valid codeword, we calculate the estimated posterior probabilities for each bit, make a hard decision and check the syndrome s . The first bit has extrinsic LLRs from the first and second checks and an intrinsic LLR from the channel the total LLR is their sum:

$$L_1 = L_1 + E_{1,1} + E_{2,1} = 9.9115 + 3.1542 + 2.5237 = 15.5894.$$

Thus even though the LLR from the channel is negative, indicating that the first bit is a 1, both the extrinsic LLRs are positive, indicating that the bit is 0. The extrinsic LLRs are large enough to make the total LLR positive and so the decision on the first bit has effectively been changed. Repeating for the second to sixth bits gives:

$$\begin{aligned}L_2 &= L_2 + E_{1,2} + E_{3,2} = -3.6830 - 4.0402 - 7.6737 = -15.3969, \\L_3 &= L_3 + E_{1,3} + E_{4,3} = -4.0430 - 3.681 - 2.5295 = -10.2535, \\L_4 &= L_4 + E_{2,4} + E_{3,4} = -7.6738 - 2.5289 - 3.6830 = -13.8857, \\L_5 &= L_5 + E_{2,5} + E_{4,5} = -2.5295 - 7.5724 - 4.0430 = -14.1449, \\L_6 &= L_6 + E_{3,6} + E_{4,6} = 16.7415 + 3.6647 - 4.0430 = 16.3632.\end{aligned}$$

The hard decision on the LLRs gives

$$\hat{c} = [0 \ 1 \ 1 \ 1 \ 1 \ 0].$$

To check whether \hat{c} is a valid codeword, consider

$$s = \hat{c}H' = [0 \ 1 \ 1 \ 1 \ 1 \ 0] \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} = [0 \ 0 \ 0 \ 0]$$

The decoding stops because $s = 0$ and the returned c is a valid code word.

8.7.3 Sum Product and Min-Sum Algorithms for Decoding of Rate 1/2 LDPC Codes Using MATLAB

The following MATLAB program and functions are written and used to decode the rate 1/2 LDPC codes using sum product and Min sum algorithms for different SNRs.

Program 8.7 MATLAB program for LDPC decoding using log domain sumproduct algorithm.

```
% sum product algorithm for LDPC decoding
% rx      : Received signal vector (column vector)
% H       : LDPC parity check matrix
% N0     : Noise variance
% iter   : Number of iteration
% xHat   : Decoded vector (0/1)
clear all;clc;
x=[ 0 1 1 1 1 0 ];
H=[1 1 1 0 0 0;1 0 0 1 1 0;0 1 0 1 0 1 ;0 0 1 0 1 1];
[N1 N2] = size(H);
algtype= input('enter 1 forlogdomain, 2 for minsum');
EbN0 = input(' Please enter Eb/N0 in dB : default [2.0]');
N0 = 1/(exp(EbN0*log(10)/10));
bpskMod = 2*x - 1;
rx = bpskMod+ sqrt(N0)*randn(size(bpskMod));
if(algtype==1)
    Li =(-4*rx./N0);
elseif(algtype==2)
    Li ==-rx;
end
iter= input('enter the number of iterations=');
if(algtype==1)
    [Eji xHat]=logsumproduct(Li,H,N1,N2,iter);
elseif(algtype==2)
    [Eji xHat]=minsum (Li,H,N1,N2,iter);
end
```

MATLAB function program logsumproduct used in Program 8.7

```
function cHat=logsumproduct(Li,H,N1,N2,iter);
Eji = zeros(N1, N2);
Pibetaji = zeros(N1, N2);
Mji = H.*repmat(Li, N1, 1);
[row, col] = find(H);% Get non-zero elements
for n = 1:iter
    fprintf('Iteration : %d\n', n);
    %ssrep2 of sumproduct algorithm in log domain
    alphaji = sign(Mji); % Get the sign and magnitude of Mji
    betaji = abs(Mji);
    for l = 1:length(row)
        Pibetaji(row(l), col(l)) = log((exp(betaji(row(l), col(l))) + 1)/...
                                         (exp(betaji(row(l), col(l))) - 1));
    end
```

```

for i = 1:N1
    c1 = find(H(i, :));
    for k = 1:length(c1)
        Pibetaji_sum = 0;
        alphaji_prod = 1;
        Pibetaji_sum = sum(Pibetaji(i, c1)) - Pibetaji(i, c1(k));
        % Avoid division by zero/very small number, get
        Pi(sum(Pi(betaij)))
            if Pibetaji_sum < 1e-20
                Pibetaji_sum = 1e-10;
            end
        Pi_Pibetaji_sum = log((exp(Pibetaji_sum) + 1)/(exp(Pibetaji_sum) - 1));
        alphaji_prod = prod(alphaji(i, c1))*alphaji(i, c1(k));
        Eji(i, c1(k)) = alphaji_prod *Pi_Pibetaji_sum;
    end % for k
end % for i
for j = 1:N2
    r1 = find(H(:, j));
    %step 3 of sumproduct algorithm in log domain
    Litotal = Li(j) + sum(Eji(r1, j));
    % step 4 of sumproduct algorithm in log domain
    if Litotal < 0
        cHat(j) = 1;
    else
        cHat(j) = 0;
    end
    %step5 of sumproduct algorithm in log domain
    for k = 1:length(r1)
        Mji(r1(k), j) = Li(j) + sum(Eji(r1, j)) - Eji(r1(k), j);
    end % for k
end % for j
cs = mod(cHat*H',2);
if sum(cs)== 0
    break;
end
end % for n

```

Step 2 of min sum algorithm

```
% step 2 of slgorithm
alphaij = sign(Mji); % Get the sign and magnitude of L(qij)
betaij = abs(Mji);
% ----- Horizontal step -----
for i = 1:N1
    % Find non-zeros in the column
    c1 = find(H(i, :));
    % Get the minimum of betaij
    for k = 1:length(c1)
        % Minimum of betaij\c1(k)
        minOfbetaij = realmax;
        for l = 1:length(c1)
            if l ~= k
                if betaij(i, c1(l)) < minOfbetaij
                    minOfbetaij = betaij(i, c1(l));
                end
            end
        end % for l
        Mji= prod(alphaij(i, c1))*alphaij(i, c1(k));
        Eji(i, c1(k)) = Mji*minOfbetaij;
    end % for k
end % for i
```

The MATLAB function minsum is same as the log sum product function program with the Step 2 in logsumproduct is replaced by the following MATLAB program segment has yielded.

For example, consider the parity check matrix of Example 8.11. $c = [0 \ 1 \ 1 \ 1 \ 1 \ 0]$ is a valid code word for the parity check matrix. When this code word is sent over an AWGN channel at $\frac{E_b}{N_o} = 2$ dB, decoding of the received vector using the above MATLAB program and functions has yielded $\hat{c} = [0 \ 1 \ 1 \ 1 \ 1 \ 0]$.

8.8 EXIT Analysis of LDPC Codes

8.8.1 Degree Distribution

An irregular parity check matrix of LDPC codes has columns and rows with varying weights i.e., a Tanner graph has bit nodes and check nodes with varying degrees. Let D_v be the number of different variable node degrees, D_c be the number of different check node degrees. Then, the following functions can be defined

$$\lambda(x) = \sum_{i=2}^{D_v} \lambda_i x^{i-1} = \lambda_2 x + \lambda_3 x^2 + \cdots + \lambda_{D_v} x^{D_v-1} \quad (8.50)$$

$$\rho(x) = \sum_{i=2}^{D_c} \rho_i x^{i-1} = \rho_2 x + \rho_3 x^2 + \cdots + \rho_{D_c} x^{D_c-1} \quad (8.51)$$

where λ_i is the fraction of edges that are connected to degree- i variable (bit) nodes, and ρ_i is the fraction of edges that are connected degree- I check nodes. It should be noted that λ_i and ρ_i must satisfy that

$$\sum_{i=2}^{D_v} \lambda_i = 1 \quad \text{and} \quad \sum_{i=2}^{D_c} \rho_i = 1$$

The code rate can be expressed as

$$r = 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx} \quad (8.52)$$

Example 8.14 Find degree distribution of the following irregular code parity check matrix

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

Solution λ_2 = the fraction of edges connected to degree 2 bit nodes = $\frac{16}{22} = 0.7273$

λ_3 = the fraction of edges connected to degree 3 bit nodes = $\frac{6}{22} = 0.2727$

ρ_4 = the fraction of edges connected to degree 4 check nodes = $\frac{12}{22} = 0.5455$

ρ_5 = the fraction of edges connected to degree 5 check nodes = $\frac{10}{22} = 0.4545$

Thus, the irregular code has degree distribution

$$\lambda(x) = 0.7273x + 0.2727x^2$$

$$\rho(x) = 0.5455x^4 + 0.4545x^5$$

8.8.2 Ensemble Decoding Thresholds

The decoding threshold in terms of the noise standard deviation (σ^*) of a given degree distribution for iterative sum product or Min Sum decoding is defined as the supreme of the channel noises for which the probability of decoding error goes to zero as the number of iterations tends to infinity. Thus, the threshold can be expressed as

$$\sigma^* = \sup \left\{ \sigma > 0 : \lim_{i \rightarrow \infty} p_b^i(\sigma) = 0 \right\} \quad (8.53)$$

Table 8.1 Thresholding values on binary AWGN channel for various regular code parameters

d_v	d_c	Rate	σ^*	$\left(\frac{E_b}{N_o}\right)^* \text{ dB}$	σ_{\max}	$\left(\frac{E_b}{N_o}\right)_{\max} \text{ dB}$
3	6	0.5	0.88	1.1103	0.979	0.1843
3	5	0.4	1.0	0	1.148	-1.1988
3	4	0.25	1.26	-2.0074	1.549	-3.8010
4	8	0.5	0.83	1.6184	0.979	0.1843
4	6	0.333	1.01	-0.0864	1.295	-2.2454
5	10	0.5	0.79	2.0475	0.979	0.1843

If $\sigma \leq \sigma^*$, $p_b^i(\sigma)$ converges to zero, otherwise converges to a value greater than zero.

The stability condition for AWGN channel is given by [11]

$$\lambda'(0)\rho'(1) < \exp\left(\frac{1}{2\sigma^2}\right) \quad (8.54)$$

Whereas the stability condition for uncorrelated Rayleigh fading channel with SI is given by

$$\lambda'(0)\rho'(1) < 1 + \frac{1}{2\sigma^2} \quad (8.55)$$

The threshold value σ^* and the maximum allowed value σ_{\max} and the corresponding $\frac{E_b}{N_o}s$ on the binary AWGN channel for various regular code parameters are given in Table 8.1 [12].

8.8.3 EXIT Charts for Irregular LDPC Codes in Binary Input AWGN Channels

Under the consistent–Gaussian assumption, the mutual information ($I_{A,V}$) between the VN (a priori) inputs and the code bit associated with that VN can be computed by using the following approximation [11] for Eq. (6.20b). Thus,

$$I_{A,V} = J(\sigma) \\ = \begin{cases} -0.0421061\sigma^3 + 0.209252\sigma^2 - 0.00640081\sigma & 0 \leq \sigma < 1.6363 \\ 1 - \exp\left(0.00181491\sigma^3 - 0.142675\sigma^2 - 0.0822054\sigma + 0.0549608\right) & 1.6363 \leq \sigma < 10 \\ 1 & \sigma \geq 10 \end{cases} \quad (8.56)$$

The approximation for inverse function $\sigma = J^{-1}(I_{A,V})$ is

$$\begin{aligned}\sigma &= J^{-1}(I_{A,V}) \\ &\approx \begin{cases} 1.09542 I_{A,V}^2 + 0.214217 I_{A,V} + 2.33727 \sqrt{I_{A,V}} & 0 \leq I_{A,V} \leq 0.3646 \\ -0.706692 \log_e 0.386013(1 - I_{A,V}) - 1.75017 I_{A,V} & 0.3646 < I_{A,V} \end{cases} \quad (8.57)\end{aligned}$$

Using $J(\sigma)$, the EXIT chart of an irregular code $I_{E,V}$ describing the variable node function can be computed as follows.

$$I_{E,V} = \sum_{i=2}^{D_v} \lambda_i J\left(\sqrt{(i-1)[J^{-1}(I_{A,V})]^2 + \sigma_{ch}^2}\right) \quad (8.58)$$

where i is the variable node degree, $I_{A,V}$ is the mutual information of the message entering the variable node with the transmitted codeword, $\sigma_{ch}^2 = 8R \frac{E_b}{N_0}$.

The EXIT chart of an irregular code $I_{E,C}$ describing the check node function can be computed as follows.

$$I_{E,C} = \sum_{i=2}^{D_c} \rho_i \left(1 - J\left(\sqrt{(i-1)} J^{-1}(1 - I_{A,C})\right)\right) \quad (8.59)$$

where i is the check node degree, $I_{A,C}$ is the mutual information of the message entering the check node with the transmitted codeword.

In order for the decoding to converge to a vanishingly small probability of error, the EXIT chart of the variable nodes has to lie above the inverse of the EXIT chart for the check nodes.

Example 8.15 Consider the following rate 1/2 irregular LDPC codes with good degree distributions for a binary AWGN channel given in [13]

Code 1:

$$\begin{aligned}\lambda(x) &= 0.33241x + 0.24632x^2 + 0.11014x^3 + 0.31112x^5 \\ \rho(x) &= 0.76611x^5 + 0.234389\end{aligned}$$

with a decoding EXIT threshold of $\frac{E_b}{N_o} = 0.6266$ dB

Code 2:

$$\begin{aligned}\lambda(x) &= 0.19606x + 0.24039x^2 + 0.00228x^5 + 0.05516x^6 \\ &\quad + 0.16602x^7 + 0.04088x^8 + 0.01064x^9 + 0.00221x^{27} + 0.28636x^{29} \\ \rho(x) &= 0.00749 + 0.99101x^8 + 0.0015\end{aligned}$$

with a decoding EXIT threshold of $\frac{E_b}{N_o} = 0.2735$ dB.

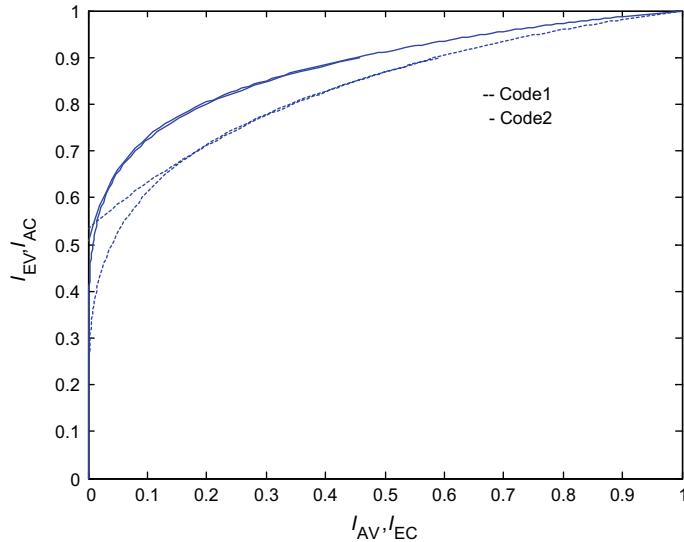


Fig. 8.7 EXIT charts for two irregular LDPC codes with different degree distributions on a Binary AWGN Channel

The EXIT charts of the two codes are shown in Fig. 8.7. From Fig. 8.7, it can be observed that the code 2 with lower threshold has better fit between variable node and check node EXIT curves.

The good degree distributions for rate 1/2 and 1/3 irregular LDPC codes for uncorrelated Rayleigh fading channels can be found in [14].

8.9 Performance Analysis of LDPC Codes

8.9.1 *Performance Comparison of Sum-Product and Min-Sum Algorithms for Decoding of Regular LDPC Codes in AWGN Channel*

The BER performance of the Sum-Product and Min-Sum LDPC decoding algorithms is evaluated through a computer simulation assuming that the channel adds white Gaussian noise to the code generated by a (256, 3, 6) regular parity check matrix. In this simulation, four hundred frames of each of length 256 and three iterations are used. The BER performance of the Sum-Product and Min-Sum Algorithms is shown in Fig. 8.8.

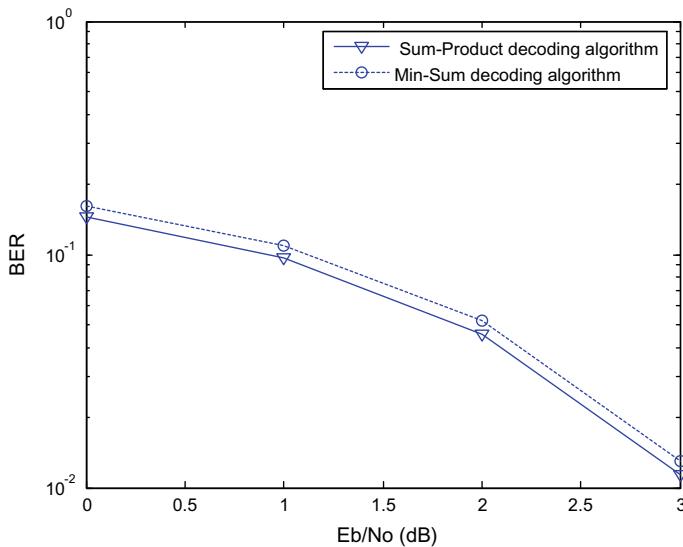


Fig. 8.8 BER performance of Sum-Product and Min-Sum decoding algorithms

8.9.2 BER Performance Comparison of Regular and Irregular LDPC Codes in AWGN Channel

The performance of rate 1/2 regular and irregular codes having the same length are evaluated through a computer simulation. The BER performance of the two codes is shown in Fig. 8.9.

From Fig. 8.9, it is observed that there is no significant difference between the BER performance of the Sum-Product and the Min-Sum algorithms.

The irregular codes can have improved thresholds for long codes but with an error floor at higher BER than for regular codes of the same rate and length.

8.9.3 Effect of Block Length on the BER Performance of LDPC Codes in AWGN Channel

The effect of block length on the performance of LDPC codes is illustrated through a computer simulation. In this experiment, two 1/2 rate irregular codes of block lengths 256, and 512 are considered and added white Gaussian noise to them, and the noisy codes are decoded using Min-Sum decoding algorithm with 10 iterations. The BER performance of the two codes is shown in Fig. 8.10.

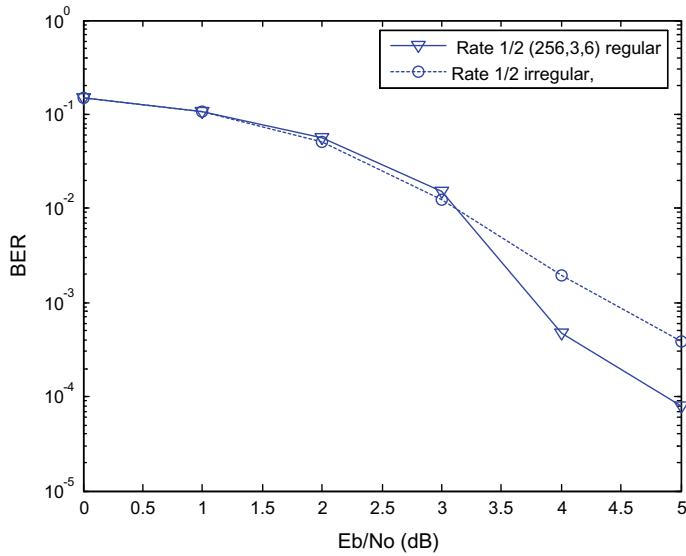


Fig. 8.9 BER performance of rate 1/2 regular and irregular LDPC codes using Min-Sum decoding algorithms

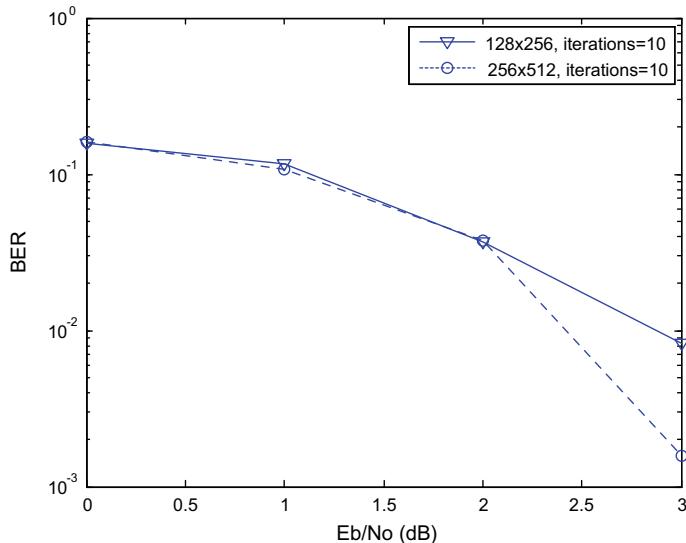


Fig. 8.10 BER performance of two 1/2 rate irregular LDPC codes using Min-Sum logarithm for decoding in AWGN Channel

8.9.4 Error Floor Comparison of Irregular LDPC Codes of Different Degree Distribution in AWGN Channel

The error-floor of an LDPC code is characterized by the phenomenon that as the SNR continues to increase, the error probability suddenly drops at a rate much slower than that in the region of low to moderate SNR can be approximated by Johnson and Weller [15]

$$\text{BER}_{ef} \approx \frac{2}{N} \frac{(\lambda_2 \rho(1)')^2}{4} Q\left(\sqrt{\frac{4RE_b}{N_o}}\right) \quad (8.60)$$

with the constraint $\lambda_2 \rho(1)' \leq E \exp\left(\frac{1}{2\sigma^2}\right)$, where E varies from 0 to 1, $E = 1$ for the traditional optimized degree distributions, E is greater than zero but less than 1 for constrained degree distributions, N is the length of the code and R is the code rate. A trade-off between the threshold and error floor can be achieved with the constrained distributions.

Example 8.16 Consider the following rate 1/4 irregular LDPC codes with optimal degree distribution and constrained degree distributions given in [15].

Code 1: Traditional code with optimal degree distribution with $E = 1$

$$\begin{aligned} \lambda(x) &= 0.431x + 0.2203x^2 + 0.0035x^3 + 0.0324x^5 + 0.1587x^6 + 0.1541x^9 \\ \rho(x) &= 0.0005x^2 + 0.9983x^3 + 0.0012x^4 \end{aligned}$$

Code 2: Code with constrained degree distribution with $E = 0.19$

$$\begin{aligned} \lambda(x) &= 0.0872x + 0.865x^2 + 0.0242x^3 + 0.0032x^5 + 0.0027x^6 + 0.0127x^9 \\ \rho(x) &= 0.0808x^2 + 0.8945x^3 + 0.0247x^4 \end{aligned}$$

Code 3: Code with constrained degree distribution with $E = 0.02$

$$\begin{aligned} \lambda(x) &= 0.0086x + 0.9711x^2 + 0.0006x^3 + 0.0059x^5 + 0.011x^6 + 0.0028x^9 \\ \rho(x) &= 0.0118x^2 + 0.9332x^3 + 0.055x^4 \end{aligned}$$

The error floor BER of the three codes is evaluated using Eq. (8.60) and shown in Fig. 8.11 along with the error floor region.

From Fig. 8.11, it can be observed that the codes with constrained degree distributions have yielded improved error floor performance. It indicates that a balance between threshold and error floor BER can be obtained.

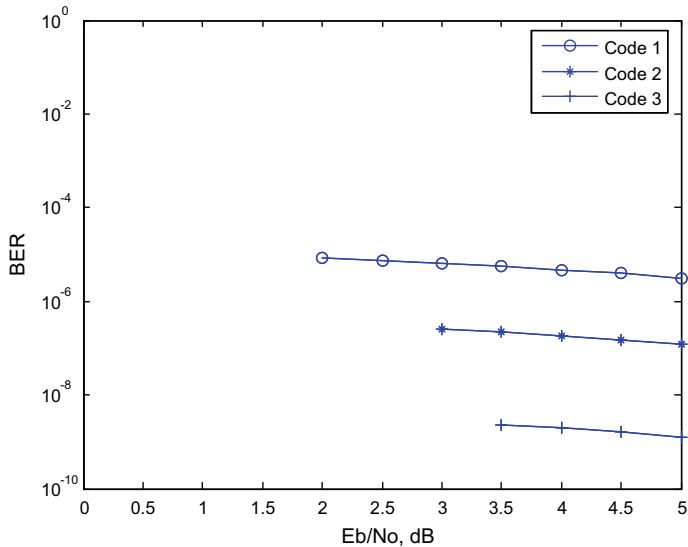


Fig. 8.11 The error floor BER of the three codes

8.10 Quasi Cyclic (QC)-LDPC CODES

The principal property of QC-LDPC codes is that their parity check matrix consists of circulant sub matrices, which could be either based on the identity matrix or a smaller random matrix. The main advantage of this construction principle compared to randomly constructed codes is that QC-LDPC encoding procedure is easier to implement. The encoder of QC-LDPC codes can be implemented by using a series of shift registers, which allows its complexity to be proportional to code length. QC-LDPC codes have been used widely in high-throughput systems, such as IEEE802.16e, IEEE802.11n, and IEEE802.11ac, and IEEE802.11ad. QC-LDPC codes are very suitable for high-throughput and low-latency system.

8.10.1 Brief Description of QC-LDPC Codes

Let $H_{M \times N}$ be the parity check matrix of a QC LDPC code. The $H_{M \times N}$ matrix of consists of m_b rows and n_b columns of submatrices of size $z \times z$, where z is a block size, $M = (m_b \times z)$ and $N = (n_b \times z)$. A submatrix is either a null matrix, or a circulant matrix obtained by shifting cyclically each row of an identity matrix to the right for p steps. Define

$$P_z \triangleq \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

then, a submatrix of size $z \times z$ with shift step p , $0 \leq p \leq z - 1$, is $(P_z)^p$, the p th power of P_z . Conventionally, $(P_z)^0$ is defined as identity matrix.

A trivial example for the parity check matrix H with $M = 6$, $N = 12$, $z = 3$, $m_b = 2$, $n_b = 4$, which is portioned into 8 sub matrices is as shown

0	1	0	0	0	0	1	0	0	0	1
0	0	1	0	0	0	0	0	1	1	0
1	0	0	0	0	0	1	0	0	0	1
1	0	0	0	1	0	0	0	1	0	0
0	1	0	0	0	1	1	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0

8.10.2 Base Matrix and Expansion

Parity check matrix of a QC-LDPC code is often described by a $m_b \times n_b$ base matrix $B_{m_b \times n_b}$. The (i, j) -th entry of $B_{m_b \times n_b}$, denoted by $b(i, j)$ is

$$\begin{aligned} b(i, j) &= -1 \text{ if } H_{i,j} = 0_{z \times z} \\ b(i, j) &= p \text{ if } H_{i,j} = (P_z)^p \end{aligned}$$

Example 8.17 Obtain base matrix for the parity check matrix shown in Sect. 8.10.1.

Solution The base matrix for the H matrix obtained in Example 8.17 is as follows

1	-1	1	2
0	1	2	-1

Given a base matrix, a parity check matrix for a code of block size z is obtained by setting submatrix as

$$\begin{aligned} H_{i,j} &= 0_{z \times z} && \text{if } b(i, j) = -1 \\ H_{i,j} &= (P_z)^{b(i,j)\bmod z} && \text{otherwise} \end{aligned}$$

Therefore, a base matrix and an expand factor z together can be used to represent H . As we can see, the information block size $K = N - M$ and N is the codeword block size. Through changing the expand factor z , a LDPC set of variable information length and certain code rate can be obtained.

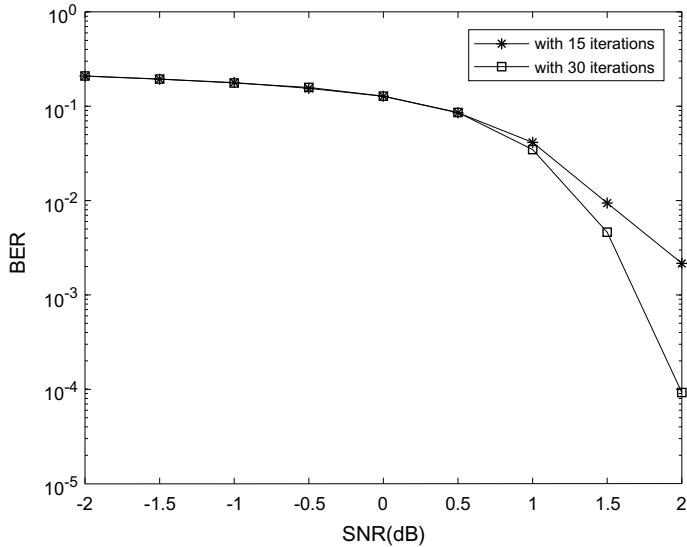
8.10.3 Performance Analysis of QC-LDPC Codes Over AWGN Channel

The performance of QC-LDPC codes is illustrated through LDPC-IEEE 802.11n. The following base matrix for $z = 27$ (rate = 1/2 mb = 12, nb = 24) specified in IEEE 802.11n standard is used in this illustration

Base_matrix = ...

$$\left[\begin{array}{cccccccccccccccccccccc} 0 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & 0 & -1 & -1 & 0 & 1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 22 & 0 & -1 & -1 & 17 & -1 & 0 & 0 & 12 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 6 & -1 & 0 & -1 & 10 & -1 & -1 & -1 & 24 & -1 & 0 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 2 & -1 & -1 & 0 & 20 & -1 & -1 & -1 & 25 & 0 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 23 & -1 & -1 & -1 & 3 & -1 & -1 & -1 & 0 & -1 & 9 & 11 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 24 & -1 & 23 & 1 & 17 & -1 & 3 & -1 & 10 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 25 & -1 & -1 & -1 & 8 & -1 & -1 & -1 & 7 & 18 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 13 & 24 & -1 & -1 & 0 & -1 & 8 & -1 & 6 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 7 & 20 & -1 & 16 & 22 & 10 & -1 & -1 & 23 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 11 & -1 & -1 & -1 & 19 & -1 & -1 & -1 & 13 & -1 & 3 & 17 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ 25 & -1 & 8 & -1 & 23 & 18 & -1 & 14 & 9 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ -3 & -1 & -1 & 16 & -1 & -1 & 2 & 25 & 5 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \end{array} \right];$$

The QC-LDPC code is generated using encoding method reported in [16]. The decoding of QC-LDPC codes is performed based on High-Throughput LDPC Decoders [17]. The following MATLAB Program 8.8 and a function program are written and executed for BER performance evaluation of QC-LDPC code with the above mentioned base matrix. The SNR versus BER performance with 15 and 30 iterations is shown in Fig. 8.12.

**Fig. 8.12** BER performance of QC-LDPC IEEE 802.11n**Program 8.8** MATLAB program for BER performance of QC-LDPC IEEE 802.11n

```
%BER performance of QC-LDPC IEEE 802.11n
clear;close all;SNR = [-2:0.5:5];itermax = 30; % max number of iterations per message
blocks = 100; MAX_R = 1000;Z = 27;
Base_matrix = ...
[0 -1 -1 -1 0 0 -1 -1 0 -1 -1 0 1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1;
22 0 -1 -1 17 -1 0 0 12 -1 -1 -1 -1 0 0 -1 -1 -1 -1 -1 -1 -1 -1;
6 -1 0 -1 10 -1 -1 -1 24 -1 0 -1 -1 -1 0 0 -1 -1 -1 -1 -1 -1 -1;
2 -1 -1 0 20 -1 -1 -1 25 0 -1 -1 -1 -1 0 0 -1 -1 -1 -1 -1 -1 -1;
23 -1 -1 -1 3 -1 -1 -1 0 -1 9 11 -1 -1 -1 -1 0 0 -1 -1 -1 -1 -1 -1;
24 -1 23 1 17 -1 3 -1 10 -1 -1 -1 -1 -1 -1 -1 0 0 -1 -1 -1 -1 -1;
25 -1 -1 -1 8 -1 -1 -1 7 18 -1 -1 0 -1 -1 -1 -1 -1 0 0 -1 -1 -1 -1;
13 24 -1 -1 0 -1 8 -1 6 -1 -1 -1 -1 -1 -1 -1 -1 0 0 -1 -1 -1;
7 20 -1 16 22 10 -1 -1 23 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0 -1 -1;
11 -1 -1 -1 19 -1 -1 -1 13 -1 3 17 -1 -1 -1 -1 -1 -1 -1 0 0 -1;
25 -1 8 -1 23 18 -1 14 9 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0;
3 -1 -1 16 -1 -1 2 25 5 -1 -1 1 -1 -1 -1 -1 -1 -1 -1 -1 0];
H = zeros(size(Base_matrix)*Z);P0 = eye(Z);
for r=1:size(Base_matrix,1)
for c=1:size(Base_matrix,2)
shift =Base_matrix(r,c);
if (shift > -1)
Pi = circshift(P0,[0 shift]);
else
Pi = zeros(Z);
end
```

```

R = (r-1)*Z+1:r*Z;C = (c-1)*Z+1:c*Z; H(R,C) = Pi;
end
[m, n] = size(H);
MAX_j = max(sum(H,2)); % max degree for check nodes
Q = zeros(1,n); % total bit-check messages
P = zeros(1,n); % temporary buffer for total bit-check messages
R = zeros(m,MAX_j); % check-bit messages
R_1 = zeros(size(R)); % previous check -bit messages
for ii = 1:length(SNR)
    noisevar = 10.^(-SNR(ii)/10);
    v = 0; Q = zeros(1,n); P = zeros(1,n);% temporary buffer for total bit-check messages
    R = zeros(m,MAX_j); % check-bit messages
    R_1 = zeros(size(R)); % previous check -bit messages
    bit_errors = 0;
    for b = 1:blocks
        if mod(b,blocks/10) == 0
            fprintf('%d%\n', b(blocks * 100));
        end
        u = randi([0 1],1,m);% Create a random word
        c = LDPC_generator(H,u);
        s = 2*c-1; % BPSK signal
        r = s + randn(size(s))*sqrt(noisevar); % received signal
        % Prior log-likelihood
        Lci = (-2*r./noisevar);P = Lci; Q = Lci; k = 0;
        while k < itermax
            for i = 1:m % for every check node
                Vi = find(H(i,:)); % Set of incident variables
                z = ones(length(Vi))-eye(length(Vi));
                Rij = Q(Vi)-R_1(i,1:length(Vi)); % Incoming messages
                % avoid NaN values
                Rij(abs(Rij)<1e-8) = 1e-8;
                R(i,1:length(Vi)) = -log(tanh(z*(-
                log(tanh(abs(Rij)/2)))'/2)).*prod(sign(Rij)).*sign(Rij(1:length(Rij)))); % outgoing message
                % avoid overflow
                R(i,abs(R(i,:)) > MAX_R) = sign(R(i,abs(R(i,:)) > MAX_R))*MAX_R;
                P(Vi) = P(Vi) + R(i,1:length(Vi));
            end
            R_1 = R; Q = P; P = Lci;v = Q<0; % current word estimation
            if ~sum(mod(H*v',2)) % valid code
                break
            end
            k = k+1;
        end
        errors = sum(u~=v(1:m) ) ; bit_errors = bit_errors + errors;
    end
    BER(ii) = bit_errors/(m*blocks);
end
semilogy(SNR,BER,'-*'); xlabel('SNR(dB)'); ylabel('BER')

```

MATLAB function program LDPC_generator used in Program 8.8

```

function c = LDPC_generator(h,u)
% define sizes of message and code
mlen = size(h,1);
clen = size(h,2);
% rename variables to match those used by the author
n = clen;
m = clen - mlen;
hrow1 = h(:,end);
% find the 'gap' length
for i=1:clen
    if hrow1(i) == 1
        g = i;
        break;
    end
end
g = mlen - g;
% width and last index of submatrices a, b
wa = n-m;
wb = g;
ea = wa;
eb = wa + wb;
% extract the submatrices a, b, c, d, e and t
a = h(1:m-g,1:ea);
b = h(1:m-g,ea+1:eb);
t = h(1:m-g,eb+1:end);
c = h(m-g+1:end,1:ea);
d = h(m-g+1:end,ea+1:eb);
e = h(m-g+1:end,eb+1:end);
% calculate p1 and p2
invt = (inv(t));
et1 = -(e*invt);
phi = et1*b + d;
xtra = et1*a + c;
p1 = mod(phi*xtra*(u'),2)';
p2 = mod(invt*(a*(u') + b*(p1')),2)';
c = [u p1 p2];
% checking c*h'=0;
zero = mod(c*h',2);
if sum(zero)== 0
    % disp('ok!')
else
    disp('error')
    return
end
end

```

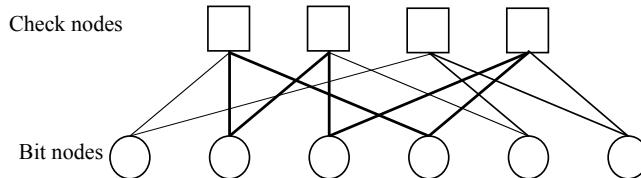
BER performance of QC-LDPC IEEE 802.11n

8.11 Problems

1. Plot the Tanner graph for the following parity check matrix H . Show that the girth of the Tanner graph is 6

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

2. Find the girth of the Tanner graph given below



3. Determine the code word for LDPC code with following parity check matrix using efficient encoding method when the message sequence $s = [1\ 0\ 0\ 0\ 0\ 0]$

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

4. A code word is generated using the following parity check matrix

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

When the code word is sent through a BEC, the received signal is

$$y = [0\ 0\ 1\ e\ e\ e]$$

Decode the received vector to recover the erased bits.

5. Consider the code word generated in Example 8.5. If it is sent through AWGN channel, the received vector after detector hard decision is $y = [1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 1]$. Decode the received vector using bit flipping algorithm and comment on the result.
6. A code word is generated with following parity check matrix

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

When the code word is sent through a BSC with crossover probability $\epsilon = 0.2$, the received signal is

$$y = [0\ 0\ 1\ 0\ 0\ 0]$$

Decode the received vector using log domain sum product algorithm.

7. Consider the code word generated in Example 8.6. If it is sent through AWGN channel with noise density $N_o = 0.3981$, the received vector is $y = [0.7271 - 2.0509 - 0.9209 - 0.8185 0.2766 - 0.2486 - 0.2497 - 1.0237 1.2065 1.1102]$. Decode the received vector using sum-product algorithm and comment on the result.
8. Repeat the problem 6 using Min-sum decoding algorithm and comment on the result.
9. Find the degree distribution of the following irregular code parity check matrix

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

10. Obtain an EXIT chart for a rate 0.49303 irregular LDPC code with the following degree distribution at $\sigma^2 = 0.97869$.

$$\lambda(x) = 0.135x + 0.2816x^2 + 0.2576x^3 + 0.0867x^{33}$$

$$\rho(x) = x^{10}$$

8.12 MATLAB Exercises

1. Write MATLAB program to generate parity check matrix H having a normalized degree distribution (from node perspective) defined as

$$\lambda(x) = \sum_{i=1}^{\lambda_{\max}} \lambda_i x^i \quad \text{and} \quad \rho(x) = \sum_{i=1}^{\rho_{\max}} \rho_i x^i,$$

with $\lambda = [0 \ 0.4994 \ 0.3658 \ 0 \ 0 \ 0 \ 0.0581 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0.0767]$;

$\rho = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1]$;

2. Write a MATLAB program to compare the performance of LDPC codes in AWGN and Rayleigh fading channels.
3. Modify MATLAB Program 8.8 to evaluate BER performance of QC-LDPC with the following base matrix and $z = 81$.

57	-1	-1	-1	50	-1	11	-1	50	-1	79	-1	1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1
3	-1	28	-1	0	-1	-1	-1	55	7	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1
30	-1	-1	-1	24	37	-1	-1	56	14	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1
62	53	-1	-1	53	-1	-1	3	35	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1
40	-1	-1	20	66	-1	-1	22	28	-1	-1	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1	-1
0	-1	-1	-1	8	-1	42	-1	50	-1	-1	8	-1	-1	-1	-1	0	0	-1	-1	-1	-1	-1
69	79	79	-1	-1	-1	56	-1	52	-1	-1	-1	0	-1	-1	-1	-1	0	0	-1	-1	-1	-1
65	-1	-1	-1	38	57	-1	-1	72	-1	27	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1	-1
64	-1	-1	-1	14	52	-1	-1	30	-1	-1	32	-1	-1	-1	-1	-1	-1	0	0	-1	-1	-1
-1	45	-1	70	0	-1	-1	-1	77	9	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1	-1	-1
2	56	-1	57	35	-1	-1	-1	-1	12	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0
24	-1	61	-1	60	-1	-1	27	51	-1	-1	16	1	-1	-1	-1	-1	-1	-1	-1	-1	0	0

References

1. R.G. Gallager, Low-density parity-check codes. IRE Trans. Inf. Theory 21–28 (1962)
2. J.L. Fan, Array codes as low-density parity-check codes, in *Proceedings of 2nd International Symposium on Turbo Codes & Related Topics*, Brest, FRANCE, 4–7 Sept 2000, pp. 543–546
3. B. Honary et al., On construction of low density parity check codes, in *2nd International Workshop on Signal Processing for Wireless Communication (SPWC 2004)*, London, U.K., 2–4 June 2004
4. B. Ammer, B. Honary, Y. Xu, S. Lin, Construction of low-density parity-check codes on balanced incomplete block designs. IEEE Trans. Inf. Theory **50**(6), 1257–1268 (2004)

5. N. Miladinovic, M. Fossorier, Systematic recursive construction of LPDC codes. *IEEE Commun. Lett.* **8**(5), 302–304 (2004)
6. S.J. Johnson, *Iterative Error Correction Turbo, Low-Density Parity-Check and Repeat-Accumulate Codes* (Cambridge University Press, 2010)
7. Y. Xiao, M.-H. Lee, Low complexity MIMO-LDPC CDMA systems over multipath channels. *IEICE Trans. Commun.* **89**(5), 1713–1717 (2006)
8. D.J.C. MacKay, R.M. Neal, Near Shannon limit performance of low density parity check codes. *Electron. Lett.* **33**, 457–458 (1997)
9. Tom Richardson, Rudiger Urbanke, Efficient encoding of low-density parity-check codes. *IEEE Trans. Inf. Theory* **47**(2), 638–656 (2001)
10. W.E. Ryan, S. Lin, *Channel Codes Classic and Modern* (Cambridge University Press, 2009)
11. S. Ten Brink, G. Kramer, Member, IEEE, A. Ashikhmin, Design of low-density parity-check codes for modulation and detection. *IEEE Trans. Commun.* **52**(4) (2004)
12. T.J. Richardson, R.L. Urbanke, The capacity of low-density parity-check codes under message-passing decoding. *IEEE Trans. Inf. Theory* **47**(2), 599–618 (2001)
13. T.J. Richardson, M. Amin Shokrollahi, R.L. Urbanke, Design of capacity-approaching irregular low-density parity-check codes. *IEEE Trans. Inf. Theory* **47**(2), 619–637 (2001)
14. J. Hou, P.H. Siegel, L.B. Milstein, Performance analysis and code optimization of low density parity-check codes on rayleigh fading channels. *IEEE J. Sel. Areas Commun.* **19**(5) (2001)
15. S.J. Johnson, S.R. Weller, Constraining LDPC degree distributions for improved error floor performance. *IEEE Commun. Lett.* **10**(2) (2006)
16. T.J. Richardson, R.L. Urbanke, Efficient encoding of low-density parity-check codes. *IEEE Trans. Inf. Theory* **47**(2), 638–656 (2001)
17. M.M. Mansour, N.R. Shanbhag, High-throughput LDPC decoders. *IEEE Trans. Very Large Scale Integr. Syst.* **11**(6), 976–996 (2003)

Chapter 9

LT and Raptor Codes



To partially compensate for the inefficiency of random codes, we can use Reed—Solomon codes, and these codes can be decoded from a block with the maximum possible number of erasures in time quadratic in the dimension. But in practice, these algorithms are often too complicated and quadratic running times are still too large for many applications. Hence, a new class of codes is needed to construct robust and reliable transmission schemes and such a class of codes is known as Fountain codes. This Fountain codes should posses a fast encoder and decoder to work in practice. Luby invented the first class of universal Fountain codes [1], in which the decoder is capable of recovering the original symbols from any set of output symbols whose size is close to optimal with high probability. The codes in this class are called LT codes. It is important to construct universal Fountain codes to many applications which have fast decoding algorithms and the average weight of an output symbol is a constant, such a class of Fountain codes are called as *Raptor codes* [2] and the basic idea behind Raptor codes is a precoding of the input symbols prior to the application of an appropriate LT code. This chapter discusses the encoding and decoding of LT and Raptor codes.

9.1 LT Codes Design

The rate less LT codes generate the limitless number of output symbols by using the encoding of a finite number of message symbols.

By receiving a given number of output symbols, each receiver can decode them successfully. The LT codes are the first universal erasure-correcting codes that provide successful communication over a binary erasure channel (BEC) for any erasure probability. The LT codes have various types of applications and advantages. In particular, a transmitter having a LT code uses a single code for efficient transmission in broadcast networks, where a single transmitter transferring a message to multiple receivers simultaneously over different channels.

The transmitter with an information message u consisting of k source symbols generates an infinite number of encoding symbols, which are broadcasted successively. Due to the property of an *ideal Fountain code*, the receiver is able to reconstruct the entire source message reliably from any k received encoding symbols. If symbols are erased, an ideal Fountain code receiver will just wait for k encoding symbols before reconstructing the information message.

In the practical implementations of LT code, random number generator was employed to determine the degree and neighbors of an encoding symbol. However, the key to make LT code work well is the degree distribution used in the encoding procedure.

9.1.1 LT Degree Distributions

Choosing a good degree distribution is the key to make LT code work well in practice and a couple of them are as follows:

A good degree distribution should meet the following two requirements: Firstly, as few encoding symbols as possible on average are required to ensure successful recovery of source symbols; secondly, the average degree of the encoding symbols shall be as low as possible.

Ideal Soliton Distribution [1]

Addition of input symbols to the ripple at the same rate as they are processed is the basic property required for a good degree distribution and hence the name soliton distribution, as a soliton wave balances dispersion and refraction perfectly. The ideal soliton distribution is given by

$$\mu_{\text{ISD}}(i) = \begin{cases} 1/k & i = 1 \\ 1/i(i-1) & i = 2, 3, \dots, k \end{cases} \quad (9.1)$$

Ideal soliton distribution ensures that at each subsequent step, all the release probabilities are identical to $1/k$. When the number of encoding symbols is equal, there is one expected ripple generated at each processing step and the source symbols can be ideally recovered after k processing step. In practice, ideal soliton distribution works poorly.

Robust Soliton Distribution [1]

Define $R_{\text{SD}} = c \cdot \ln(k/\varepsilon) \sqrt{k}$. In this distribution, we computed μ_{RSD} as follows

$$\tau(i) = \begin{cases} R_{\text{SD}}/ik & i = 1, 2, \dots, \text{round}(k/R_{\text{SD}}) - 1 \\ R_{\text{SD}} \ln(R_{\text{SD}}/\varepsilon)/k & i = \text{round}(k/R_{\text{SD}}) \\ 0 & \text{else} \end{cases} \quad (9.2)$$

$$\beta = \sum_{i=1}^k \mu_{\text{ISD}}(i) + \tau(i)$$

$$\mu_{\text{RSD}}(i) = (\mu_{\text{ISD}}(i) + \tau(i))/\beta, \quad i = 1, 2, \dots, k \quad (9.3)$$

Robust soliton distribution is an improvement of the ideal soliton distribution which is not only viable but practical too.

The term R_{SD} used for the RSD above is thought as the size of the ripple.

R_{SD} is simply understood as the number of information symbols, and at each decoding step, its degree is one. The ripple is the set of covered input symbols that have not yet been processed. By the following relation, we can predetermine this value

$$R_{\text{SD}} = c \cdot \ln(k/\varepsilon) \sqrt{k} \quad (9.4)$$

where c and ε are two parameters. c controls the mean of degree distribution and ε is the allowable failure probability of the decoder to recover the source symbols. The smaller the value of c , the greater the probability of low degrees.

In the encoding for LT codes, the degree distribution of information symbols and output symbols should be a uniform distribution and a RSD, respectively.

Luby suggested that the number of received encoded symbols be βk . Then the corresponding decoding overhead is

$$\begin{aligned} \varepsilon &= n/k - 1 \\ &= 1/R_{\text{SD}} - 1 \\ &= \beta - 1 \end{aligned} \quad (9.5)$$

The average degree increases logarithmically versus the code dimension. As the code dimension k is getting larger, the overhead decreases this is because β is a decreasing function with k , but the sparseness of the generator matrix becomes lower and lower.

9.1.2 Important Properties of the Robust Soliton Distribution

Property 1: The number of encoding symbols is $n = k + O(\sqrt{k} \cdot \ln^2(k/\varepsilon))$.

Property 2: The average degree of an encoding symbol is $D = O(\ln(k/\varepsilon))$.

Property 3: The decoder fails to recover the data with probability at most ε from a set of n encoding symbols.

One can refer [1] for proofs of these properties.

9.1.3 LT Encoder

The stepwise procedure to produce infinite output symbols from k input symbols $\{S_1, S_2, \dots, S_k\}$ is as follows:

- Step 1. Consider an output degree d randomly from a degree distribution $\rho(d)$.
- Step 2. Select d distinct input symbols uniformly at random from $\{S_1, S_2, \dots, S_k\}$.
- Step 3. Perform exclusive-OR of these d input symbols to obtain the output symbol

$$c_i = S_{i,1} \oplus S_{i,2} \oplus \dots \oplus S_{i,d}$$

A generator matrix G also can be defined such that the output symbols can be expressed as:

$$c = s \cdot G \quad (9.6)$$

where s denotes the input vector. Modulus-2 addition is used during the matrix multiplication.

The following MATLAB function generates the matrix G by using robust soliton density.

Program 9.1: MATLAB function to generate using robust soliton density

```

function G = Grsd(K, N)
if nargin < 3
    ColTH = 1.3; %bound for number of 1's in each column
end
if nargin < 4
    Num_Try = 10;
end
G_Try = {};
Mean_SR = zeros(1, Num_Try);
Min_SR = zeros(1, Num_Try);
for Ind_Try = 1: Num_Try
    while(1)
        %create pdf
        c = 0.2;
        delta = 0.5;
        R_SD = c*log(K/delta)*sqrt(K);
        rho = zeros(1, K);
        thu = zeros(1, K);
        d = 1: K;
        rho(1) = 1/K;
        rho(2:K) = 1 ./ (d(2:end).* (d(2:end)-1));
        loc = floor(K/R_SD-1);
        thu(1: loc) = R_SD/K ./ d(1: loc);
        thu(loc+1) = R_SD/K*log(R_SD/delta);
        Z = sum(rho + thu);
        mu = (rho + thu) / Z;
        if Z >= ColTH
            break;
        end
    end
    G_Try{Ind_Try} = G;
    Mean_SR = Mean_SR + G;
    Min_SR = min(Min_SR, G);
end
Mean_SR = Mean_SR / Num_Try;
Min_SR = Min_SR / Num_Try;
G = Mean_SR;

```

```
%threshold for maximum number of ones per column
MaxColTH = K/R_SD * ColTH;
G = zeros(K, N);
for n = 1: N
    while(1)
        %draw a number
        Cmu = cumsum(mu)/sum(mu);
        u = rand(1,1);
        diff = abs(u - Cmu);
        [M, loc] = min(diff);
        Num = d(loc);
        if Num <= MaxColTH
            break;
        end
    end
    row = randperm(K);
    G(row(1: Num), n) = 1;
end
SR = sum(G, 2);
if min(SR) > 0
    Mean_SR{Ind_Try} = mean(SR);
    Min_SR{Ind_Try} = min(SR);
    G_Try{Ind_Try} = sparse(G);
    break;
end
end
%maximize weight per row and minimize weight per col
[V, loc] = max(Mean_SR);
G = G_Try{loc};
G = full(G);
```

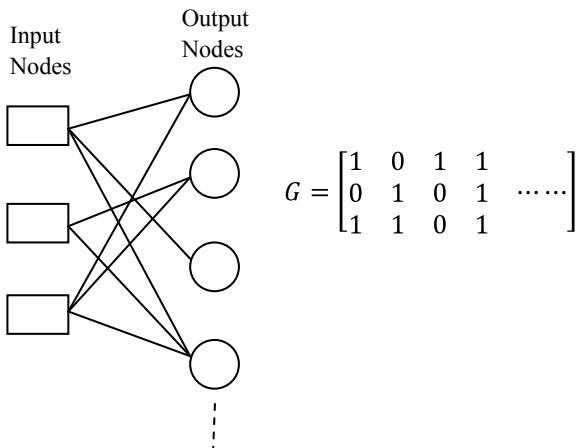
9.1.4 Tanner Graph of LT Codes

The Tanner graph of LT codes is similar to the Tanner graph used in LDPC codes, whereas the check nodes and variable nodes usually used in LDPC codes are replaced with input nodes and output nodes of LT codes as shown in Fig. 9.1.

9.1.5 LT Decoding with Hard Decision

The decoder uses the decoder recovery rule [1] to repeatedly recover input symbols. The decoder recovery rule is as follows

Fig. 9.1 Tanner graph of LT codes



If there is at least one encoding symbol that has exactly one neighbor then the neighbor can be recovered immediately since it is a copy of the encoding symbol. The value of the recovered input symbol is exclusive-ored into any remaining encoding symbols that also have that input symbol as a neighbor, the recovered input symbol is removed as a neighbor from each of these encoding symbols and the degree of each such encoding symbol is decreased by one to reflect this removal.

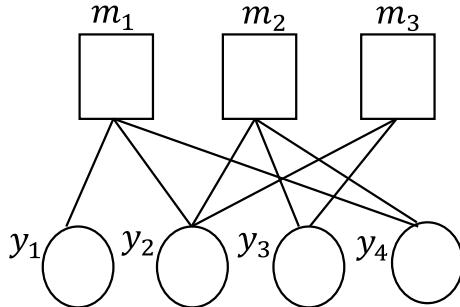
Based on the above rule, the stepwise decoding process can be described as follows:

- Step 1. Find an output symbol y_i connected to only one input symbol m_j .
- Step 2. Set $m_j = y_i$.
- Step 3. Exclusive-OR m_j to all the output symbols connected to m_j .
- Step 4. Remove all the edges connected to m_j .
- Step 5. Repeat step 1 to 4 until all input symbols are recovered.

The output degree distribution is a critical part of LT codes. If there is no output symbol with degree one during the iteration, the decoding process will halt, which indicates a decoding failure. Thus, optimal output degree is required to ensure a successful decoding. In Luby's paper, two output degree distributions, i.e., ideal soliton distribution and robust soliton distribution are presented [1].

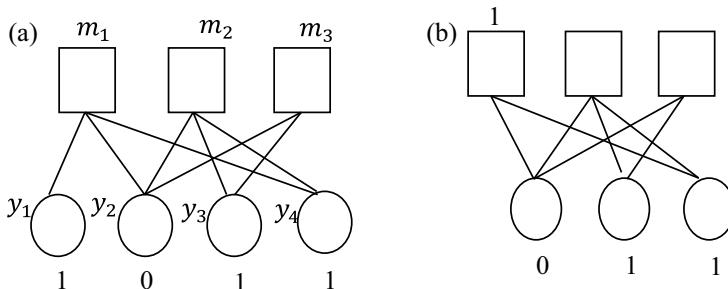
Ideal soliton distribution adds only one input node to the ripple each iteration round, consuming the fewest output symbols to recover all of the input symbols. However, it performs poorly in practice because the probability of ripple vanishing, i.e., the error probability or the decoding failure rate is high. Robust soliton distribution attempts to lower the error probability by slightly increasing the probability of degree one. The following example illustrates the decoding process.

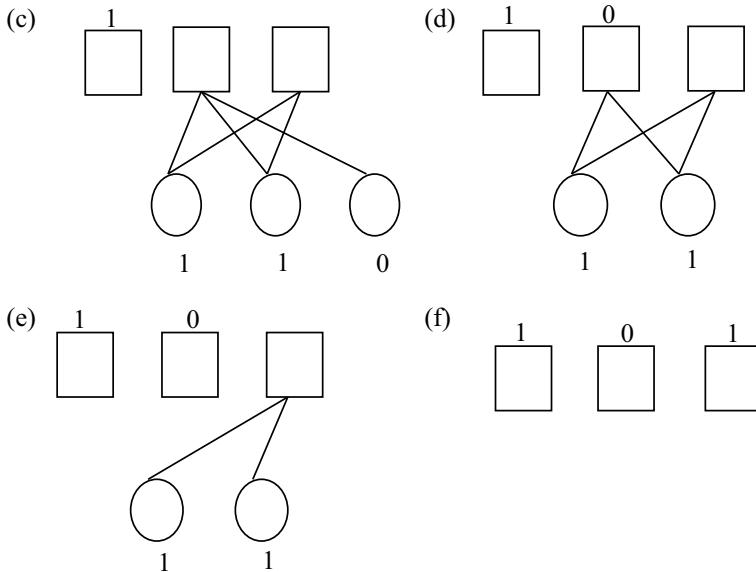
Example 9.1 Consider the following Tanner graph of a LT code



and if the received bits vector $y = [1011]$. Decode the received bits vector to obtain the message bits.

Solution The boxes represent the message bits, while circles represent output bits. An output bit is the factor graph. There are three message bits and four output bits, which have values $y_1 y_2 y_3 y_4 = 1011$. During the first iteration, the only output bit that is connected to one message bit is the first output bit (see Fig. a). This value is copied to m_1 , delete the output bit (see Fig. b), and then the new value of m_1 gets added to y_2 and y_4 . This disconnects m_1 from the graph (see Fig. c). At the start of the second iteration, y_4 is connected to the single message bit m_2 . Now, one sets m_2 equal to y_4 (see Fig. d) and then add this value to y_2 and y_3 . This disconnects m_2 from the graph (see Fig. e). Finally, one sees that the output bits connected to m_3 are equal as expected and can be used to restore m_3 (see Fig. f).





For success of the LT decoder with high probability, decoding graph of LT codes needs to have at least $k \ln(k)$ edges. In [2], Raptor codes are introduced to relax this lower bound on the number of edges in the decoding graph.

9.1.6 Hard Decision LT Decoding Using MATLAB

The following example illustrates decoding of LT codes using MATLAB.

Example 9.2 Write a MATLAB program to implement hard decision LT decoding assuming received vector $y = [1\ 0\ 1\ 1\ 0\ 1\ 1]$ when the following generator matrix is used to encode the codeword.

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Solution The following MATLAB program decodes the received vector y . The output of the program gives the decoded vector $\begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix}$.

Program 9.2: MATLAB program for Hard Decision LT Decoding

```

clear all;clc;
G=[ 1 1 0 1 0 0 0;0 0 1 1 0 0 1;1 0 0 1 1 0 0;0 0 1
0 1 1 1];
Y=[1 0 1 1 0 1 1]';
Yd=decodeLT(G,Y,1);
function Yd = decodeLT(G, Y, B)
K = size(G, 1);
N = size(G, 2);
GT = G;
Yd = -ones(K, 1);
Summer = zeros(1, K);
while(1)
    loc = find(sum(GT,1) == 1);
    if isempty(loc)
        %code fails
        break;
    end
    for ind = 1: length(loc)
        pos = find(GT(:, loc(ind)) == 1);
        loc2 = find(GT(pos, :) == 1);
        GT(pos, :) = 0;
        Summer(pos) = 1;
        Yd(pos) = Y(loc(ind));
        Y(loc2) = mod(Y(loc2)-Yd(pos), 2^(B));
    end
    if sum(Summer) == K
        break;
    end
end

```

9.1.7 BER Performance of LT Decoding Over BEC Using MATLAB

At the receiving end, a receiver collects $n = k(1 + \eta)$ output symbols. Here, η is called the overhead. Note that several erasures need to be discarded till n unerased output symbols are obtained. The following MATLAB program illustrates the BER performance of LT codes over BEC for different overheads and for different erasure probabilities.

Program 9.3: MATLAB program for the BER performance of LT decoding over BEC

```

clear all; clc;
K=1000; frame=100;
M=[500 600 700 800 900 1000];
Pe=[0.1 0.2 0.3 ];
for ii=1:length(Pe)
for i = 1:length(M )
    G=Grsd(K,(K+M(i)));
    berl(i) = 0;
    for j = 1:frame
        msg = randi(255,125,1);
        Bits = (dec2bin(msg, 8))';
        Bits = (Bits(:))';
        BitsVec = zeros(1, length(Bits));
        for Ind = 1: length(Bits)
            BitsVec(Ind) = str2double(Bits(Ind));
        end
    Y=mod(G'*BitsVec',2);
    ErrLocBit = find(rand(length(Y), 1)< Pe(ii));
    ErasureBit = zeros(length(Y),1);
    ErasureBit(ErrLocBit)= 1;
    Y(ErasureBit==1)=[];
    DecBits = decodeLT(G(:,ErasureBit==0),Y);
    DecBits(find(DecBits < 0)) = 0;
    DecBits = DecBits';
        %calculate error
    [num1,rat1]=biterr(DecBits,BitsVec);
    berl(i) = (berl(i)+rat1);
    end % for j
    berl(i) = berl(i)/frame; % Get average of BER
end % for i
ber(ii,:)=berl;
end
semilogy(M/1000, ber(1,:), '-v');
hold on
semilogy(M/1000, ber(2,:), '-d');
semilogy(M/1000, ber(3,:), '-s');
xlabel('Overhead'); ylabel('BER')
legend('Pe=0.1','Pe=0.2','Pe=0.3')

```

The overhead versus BER performance of LT decoding obtained from the above program for three different erasure probabilities is shown in Fig. 9.2.

From Fig. 9.2, it is observed that the BER decreases as overhead increases for a fixed erasure probability and further the BER for low erasure probability is less than that the high erasure probability.

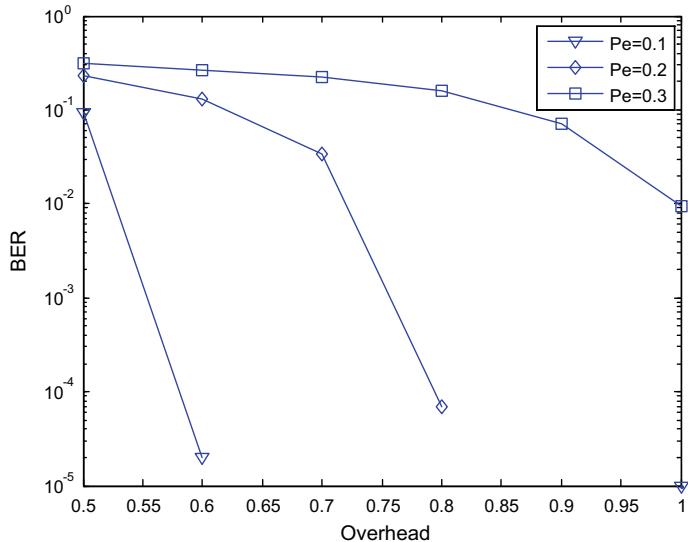


Fig. 9.2 BER performance of LT decoding over BEC

9.2 Systematic LT Codes

Many researchers endeavored to improve the performance of LT codes to protect the data over the wireless Internet, where fading, noise, and packet erasures are encountered, for the sake of improving the error correction capability of LT codes and the complexity of these schemes tends to be increased. The soft decoding of LT codes is using the probabilistic decoding technique of Low-Density Parity Check Codes (LDPC) [3].

Hence, to improve the LT code's performance in hostile wireless channels, a systematic LT code as shown in Fig. 9.3 has been suggested in [4] by expanding LT code's $[K \times N]$ generator matrix with the aid of a unity matrix having a size of $[K \times K]$.

Fig. 9.3 The systematic LT generator matrix

The relation between the systematic LT generator matrix G and the parity check matrix H is as follows:

Consider a generator matrix $G_{K \times N} = [I_{K \times K}/A_{K \times M}]$, where I is an identity matrix having a size of $[K \times K]$ and A is a non-singular matrix having a size of $[K \times M]$.

Then the parity check matrix is calculated as $H = [A^T/I']$, where A^T is the transpose of A and I' is an identity matrix having a size of $[M \times M]$ [5], where $N = K + M$ is the number of columns in G and K is the number of rows in G .

9.2.1 Systematic LT Codes Decoding

The implementation of the LT decoding process is similar to that of the classic LDPC decoding procedure. The LT decoder's soft values are set to a value corresponding to the demodulator's soft output. The decoder's soft values which denote the log-likelihood ratios (LLRs) are passed from the check nodes to the variable nodes and vice versa are then iteratively updated after each decoding iteration.

The LT decoder outputs its tentative hard decision and checks after each iteration whether the product of the corresponding codeword and the transpose of the PCM H is equal to zero, if not, the LT decoding process will be continued iteratively until the output codeword becomes legitimate or the maximum affordable number of iterations is exhausted.

9.2.2 BER Performance Analysis of Systematic LT Codes Using MATLAB

9.2.2.1 BER Performance of Systematic LT Codes Over BEC

The performance of the systematic LT (1000, 3000) code on BEC channel with erasure probabilities [0.1 0.2 0.4 0.6 0.8] is illustrated using the following MATLAB program. In this, the robust soliton degree distribution with parameters $c = 0.1$ and $\varepsilon = 0.5$ is used.

Figure 9.4 shows the performance analysis of the systematic LT code over the BEC having different erasure probabilities ρ_e .

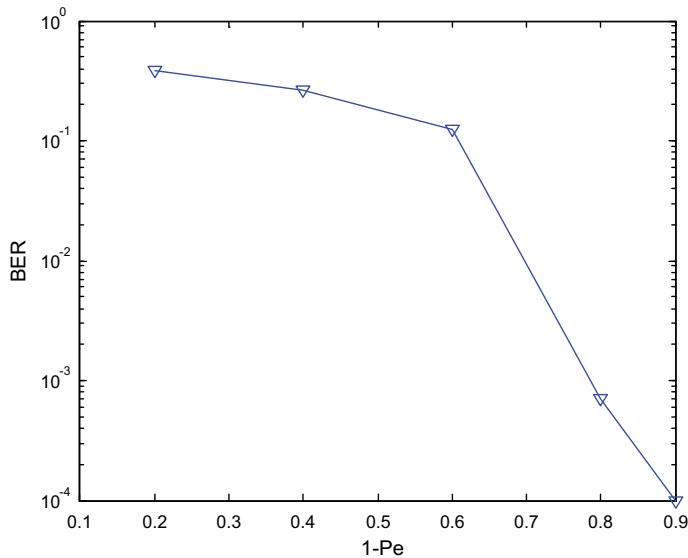


Fig. 9.4 BER versus 1-Pe performance of the systematic LT code in BEC channels

Program 9.4: MATLAB program for the BER performance of systematic LT code in BEC channels

```

clear all; clc;
K=1000;
M=1000;
Pe = [0.1 0.2 0.4 0.6 0.8 ];
frame=10;
A1=Grsd(K,M);
G1=[eye(K) A1];
H1=[ A1' eye(M)];
[N1 N2]=size(H1);
for i = 1:length(Pe )
    ber1(i) = 0;
    for j = 1:frame
        msg = randi(255,125,1);
        Bits = (dec2bin(msg, 8))';
        Bits = (Bits(:))';
        BitsVec = zeros(1, length(Bits));
        for Ind = 1: length(Bits)
            BitsVec(Ind) = str2double(Bits(Ind));
        end
    end
end

```

```

F1=mod(G1'*BitsVec', 2);
xHat1b=bec_channel(F1,Pe(i));
xHat1be=becebest(xHat1b,H1,50);
[num1, rat1] = biterr(abs(xHat1be(1:1000))',F1(1:1000)');
ber1(i) = (ber1(i) + rat1);
end % for j
% Get average of BER
ber1(i) = ber1(i)/frame;
end % for
semilogy(1-Pe, ber1, '-v');
 xlabel('1-Pe');
 ylabel('BER')

function [ y ] = bec_channel( x, e )
%BEC_CHANNEL Simulates binary erasure channel
with erasure probability e
y = x;
y( rand(size(abs(x)))<e ) =-1;% = erasure,
otherwise 0s and 1s are bits
end

```

9.2.2.2 BER Performance of Systematic LT Codes Over AWGN Channel

The performance of the systematic LT (1000, 2000) code on AWGN channel is illustrated using the following MATLAB program. In this, the robust soliton degree distribution with parameters $c = 0.1$ and $\varepsilon = 0.5$ is used.

Figure 9.5 shows the performance analysis of the systematic LT code over the AWGN having different E_b/N_0 s.

Program 9.5: MATLAB program for the BER performance of systematic LT code in AWGN channels using BPSK modulation

```

clear all; clc;
K=1000;M=1000; frame=10;
EbNO = [ 0 1 2 3 4 ];
iter=1;iter2=2;iter4=4;iter6=6;
A1=Grsd(K,M);
G1=[eye(K) A1];
H1=[ A1' eye(M) ];
[N1 N2]=size(H1);

```

```

for i = 1:length(EbN0)
    ber1(i) = 0;
    ber2(i) = 0;
    ber3(i) = 0;
    ber4(i) = 0;
for j = 1:frame
    msg = randi(255,125,1);
    Bits = (dec2bin(msg, 8))';
    Bits = (Bits(:))';
    BitsVec = zeros(1, length(Bits));
    for Ind = 1: length(Bits)
        BitsVec(Ind) = str2double(Bits(Ind));
    end
    F1=mod(G1'*BitsVec', 2);
    bpskMod1 = 2*F1 - 1;
    N0 = 1/(exp(EbN0(i)*log(10)/10));
    Ftx1 = bpskMod1 + sqrt(N0)*randn(size(bpskMod1 ));
    L1 = (-4*Ftx1./N0)';
    xHat1=logsumproduct(L1,H1,N1,N2,iter);
    xHat2=logsumproduct(L1,H1,N1,N2,iter2);
    xHat3=logsumproduct(L1,H1,N1,N2,iter4);
    xHat4=logsumproduct(L1,H1,N1,N2,iter6);
    [num1,rat1]= biterr(abs(xHat1(1:1000)'),F1(1:1000));
    [num2,rat2]= biterr(abs(xHat2(1:1000)'),F1(1:1000));
    [num3,rat3]= biterr(abs(xHat3(1:1000)'),F1(1:1000));
    [num4,rat4]= biterr(abs(xHat4(1:1000)'),F1(1:1000));
        ber1(i) = (ber1(i) + rat1);
        ber2(i) = (ber2(i) + rat2);
        ber3(i) = (ber3(i) + rat3);
        ber4(i) = (ber4(i) + rat4);
    end % for j
    % Get average of BER
    ber1(i) = ber1(i)/frame;
    ber2(i) = ber2(i)/frame;
    ber3(i) = ber3(i)/frame;
    ber4(i) = ber4(i)/frame;
end % for i
semilogy(EbN0, ber1, '-v');hold on
semilogy(EbN0, ber2, '-d');semilogy(EbN0,ber3,'-s');
semilogy(EbN0, ber4, '-*');
legend('Systematic LT(1000,2000),AWGN channel, iterations=1',...
    'Systematic LT(1000, 2000),AWGN channel, iterations=2',...
    'Systematic LT(1000, 2000),AWGN channel, iterations=4',...
    'Systematic LT(1000, 2000),AWGN channel, iterations=6')
 xlabel('Eb/N0 (dB)');
 ylabel('BER')

```

9.2.2.3 BER Performance of Systematic LT Codes Over AWGN Contaminated BEC

The schematic of the encoding and decoding of the LT coding over AWGN contaminated BEC is shown in Fig. 9.6.

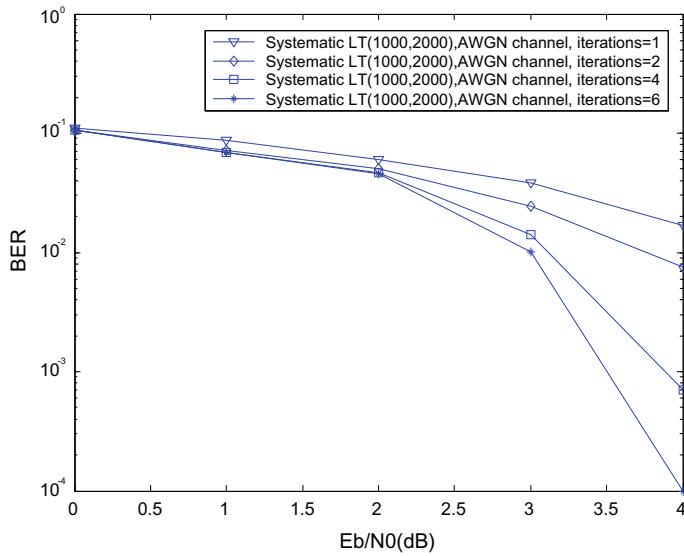


Fig. 9.5 BER versus E_b/N_0 performance of the systematic LT code in AWGN channels using BPSK modulation

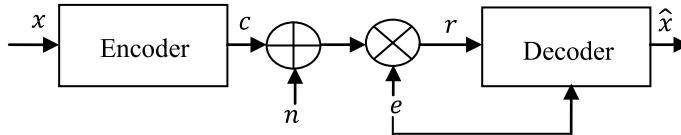


Fig. 9.6 LT encoding and decoding over AWGN–BEC

The performance of the systematic LT (1000, 3000) code on AWGN contaminated BEC with erasure probability 0.1 is shown in Fig. 9.7. In this, the robust soliton degree distribution with parameters $c = 0.1$ and $\varepsilon = 0.5$ is used.

9.3 Raptor Codes

Raptor codes are concatenation of an erasure-correcting pre-code and an LT code [2]. Figure 9.8 shows a graphical presentation of a Raptor code. The output symbols of the Raptor code are sampled independently from the distribution. Low-Density Parity Check (LDPC) codes and Tornado codes are examples of the pre-code. The decoding graph of the LT code with k -symbol message block should have at least $k \ln(k)$ edges, which results in large overhead or higher complexity of encoding and decoding. The Raptor codes reduce the lower bound on the number of edges in the bipartite graph and hence recover message symbols at a lower overhead at almost

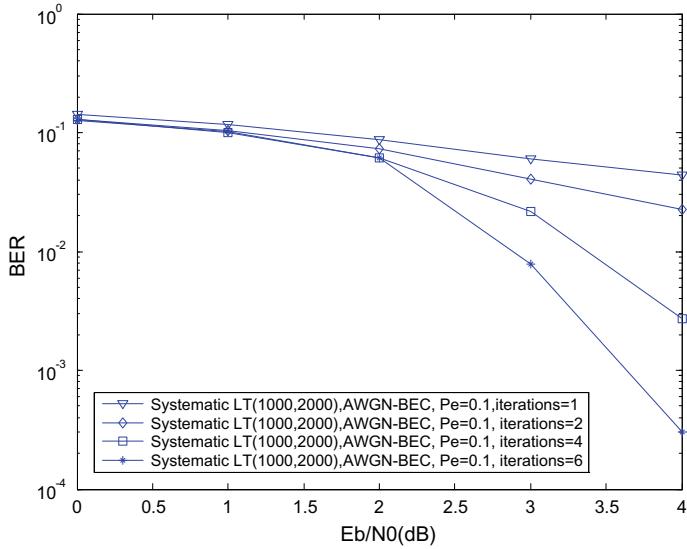


Fig. 9.7 BER versus E_b/N_0 performance of the systematic LT code in AWGN contaminated BEC channel

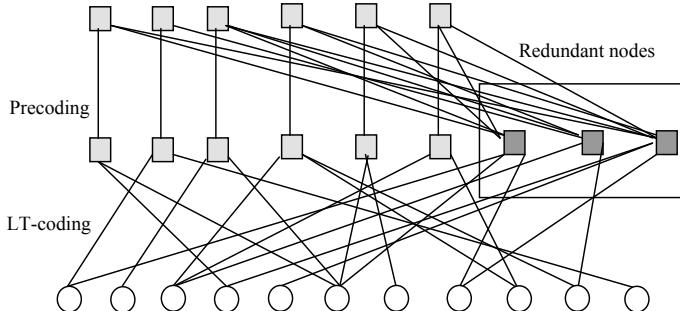


Fig. 9.8 Raptor codes

linear complexity. The pre-code of the Raptor code recovers the message symbols that are left undecoded by the LT decoders due to lower head.

Raptor codes are being used in commercial systems of Digital Fountain. For example, a Silicon Valley-based startup specializing in fast and reliable delivery of data over heterogeneous networks.

The k symbols are input symbols of a Raptor code which are used to construct the codeword in C consisting of n *intermediate symbols* and output symbols are the symbols generated by the LT code from the n intermediate symbols.

Raptor code encoding algorithm is as follows: an encoding algorithm for \mathcal{C} is used to generate a codeword in \mathcal{C} corresponding to the given k input symbols, and then an encoding algorithm for the LT code with distribution $\Omega(x)$ is used to generate the output symbols.

Raptor code decoding algorithm of length m can recover the k input symbols from any set of m output symbols and errors with probability which is at most $1/k^C$ for some positive constant.

9.4 Problems

- Suppose the generator matrix G of the LT code is

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

and the received encoded symbols are [1 0 1 1 0 0 0]. Decode the received vector to obtain message symbols.

- Show that the number of encoding symbols is $n = k + O(\sqrt{k} \cdot \ln^2(k/\varepsilon))$.
- Show that the average degree of an encoding symbol is $D = O(\ln(k/\varepsilon))$.
- Show that the decoder fails to recover the data with probability at most ε from a set of N encoding symbols.

9.5 MATLAB Exercises

- Write a MATLAB program for LT encoding and decoding with non-binary message symbols using robust soliton distribution.
- Write a MATLAB program to generate H matrix for a systematic LT code with k message symbols and 0.5 overhead.

References

- M. Luby, LT codes, in *Proceeding of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, November 2002, pp. 271–282
- A. Shokrollahi, Raptor codes. *IEEE Trans. Inf. Theory* **52**(6), 2551–2567 (2006)

3. T. J. Richardson, R. L. Urbanke, The capacity of low density parity check codes under message-passing decoding. *IEEE Trans. Inf. Theory*, 599–618 (2001)
4. T.D. Nguyen, L.-L. Yang, L. Hanzo, Systematic Luby transform codes and their soft decoding, in *IEEE SiPS'07*, Shanghai, China, 17–19 October 2007, pp. 67–72
5. R. Gallager, Low density parity check codes. *IRE Trans. Inf. Theory*, 21–28 (1962)

Chapter 10

Polar Codes



A new channel coding has flourished known as polar coding, and it is a channel coding scheme that was invented by Erdal Arikan at Bilkent University (Ankara, Turkey). Polar codes are said to achieve channel capacity in a given binary discrete memoryless channel. This can be achieved only when the block size is large enough. The complexity of encoding and decoding is less and these codes can be successfully decoded.

The main idea behind the polar codes invented by Arikan [1] is the channel polarization. The channels can be categorized into good and bad by channel polarization. The recursive application of the polarization transformation makes the synthesized channels having better reliability for good channels and worse reliability for bad channels.

The channels can be distinctly polarized with increase in code length and the information bits are transmitted over good channels, whereas the frozen bits are transmitted over bad channels.

This chapter discusses channel polarization, polar encoder structures and encoding, and polar decoding.

10.1 Channel Polarization

Polarization is an operation by which one obtains a second set of N polarized channels $\{W_N^{(i)} : 1 \leq i \leq N\}$ out of given N independent channels. For large N , the capacity term $\{I(W_N^{(i)})\}$ approaches toward 0 or 1 for all but a vanishing fraction of indices i .

The two phases in the process of channel polarization are

1. Channel combining phase.
2. Channel splitting phase.

10.1.1 Channel Combining Phase

In this phase, a vector channel $W_N : X^N \rightarrow Y^N$, with $N = 2^i, i \geq 0$, is produced by combining copies of binary discrete memoryless channel (B-DMC) W in a recursive manner. Only one copy of W makes $W_1 \triangleq W$ at zeroth level ($i = 0$) recursion (Fig. 10.1).

Two independent copies of W_1 are combined at the first-level recursion ($i = 1$) to create the channel W_2 as shown in Fig. 10.2 and obtain the channel with the transition probabilities

$$W_2(y_1, y_2|u_1, u_2) = W(y_1|u_1 \oplus u_2)W(y_2|u_2) \quad (10.1)$$

The next level of the recursion is shown in Fig. 3.3 where two independent copies of W_2 are combined to create the channel $W_4 : X^4 \rightarrow Y^4$ with transition probabilities

$$W_4(y_1^4|u_1^4) = W_2(y_1^2|u_1 \oplus u_2, u_3 \oplus u_4)W_2(y_3^4|u_2, u_4) \quad (10.2)$$

A combined channel for $N = 4$ is shown in Fig. 10.3.
 R_4 is the permutation operation that maps an input (u_1, u_2, u_3, u_4) to (u_1, u_3, u_2, u_4) .

Fig. 10.1 Parallel DMCs

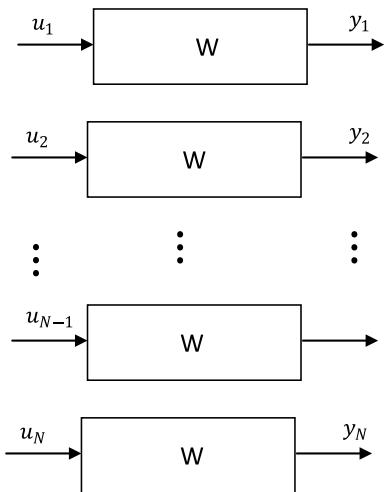
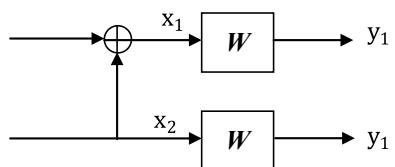


Fig. 10.2 Combined channel W_2



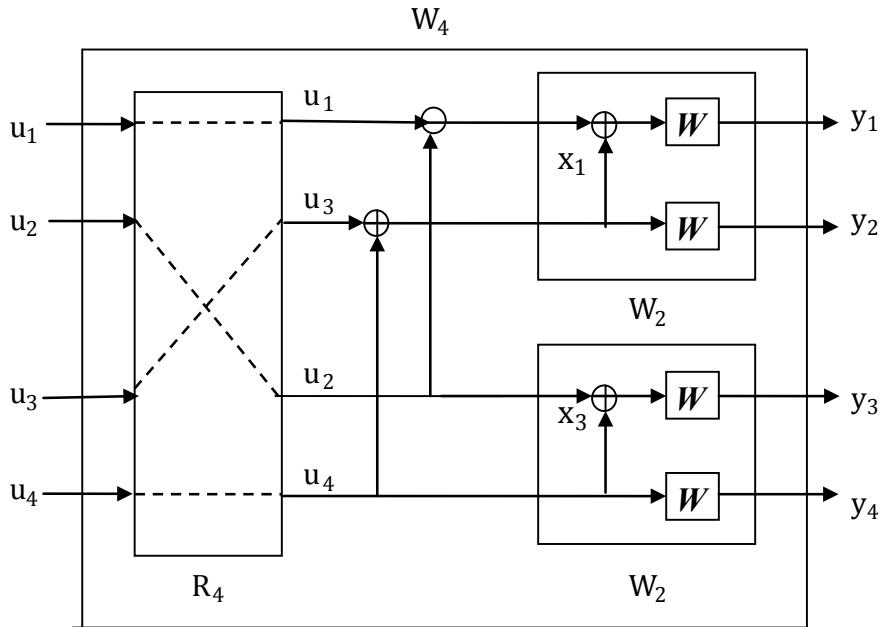


Fig. 10.3 Combined channel W_4

The combined channel W_4 shown in Fig. 10.3 can be represented in alternate form as shown in Fig. 10.4.

The Combined channel W_4 shown in Fig. 10.3 can be redrawn as shown in Fig. 10.5.

Where G_4 is the generator matrix for $N = 4$. It is given by

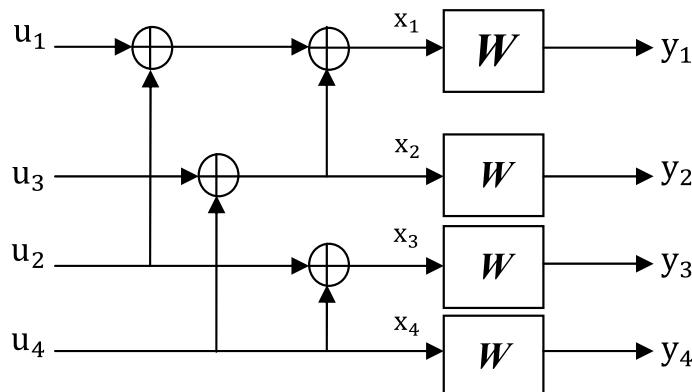


Fig. 10.4 An alternate form of combined channel W_4

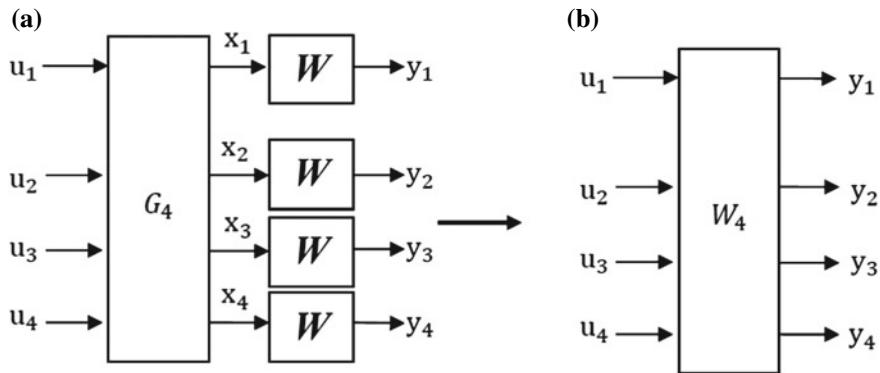


Fig. 10.5 Combined channel W_4

$$G_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Similarly, two independent copies of W_4 are combined to create the channel W_8 and so on.

A. Generator Matrix Construction

The generator matrix G_N is given by the following equation where B_N is the permutation matrix

$$G_N = B_N F^{\otimes n} \quad \{N = 2^n, n \geq 0\} \quad (10.3)$$

where

$$F \triangleq \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

$F^{\otimes n}$ is the n th Kronecker product of F .

F is the variable that is going to completely satisfy the channel combining operation.

The Kronecker product should not be confused with the usual matrix multiplication, which is an entirely different operation.

Kronecker product of an $m \times n$ matrix $A = [A_{ij}]$ and an $r \times s$ matrix $B = [B_{ij}]$ is defined as

$$A \otimes B = \begin{bmatrix} A_{11}B & \dots & A_{1n}B \\ \vdots & \ddots & \vdots \\ A_{m1}B & \dots & A_{mn}B \end{bmatrix} \quad (10.4)$$

which is an $mr \times ns$ matrix.

For $N = 2$, the generator matrix G_2 is given by

$$G_2 = B_2 F = B_2 F = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \quad (10.5)$$

Since $B_2 = I_2$, $G_2 = F$.

For $N = 4$, the generator matrix can be constructed from G_2 by constructing two matrices G_{21} and G_{22} and concatenating them columnwise as [2]

$$G_4 = \begin{bmatrix} G_{21} \\ G_{22} \end{bmatrix} \quad (10.6)$$

If G_2 is represented as

$$G_2 = [C_1 \ C_2] \quad (10.7)$$

where C_1 and C_2 stand for columns 1 and 2 of G_2 .

Then G_{21} is obtained by inserting zero column vectors as

$$G_{21} = [C_1 \ 0 \ C_2 \ 0] \quad (10.8)$$

G_{22} is obtained by duplicating the columns as

$$G_{22} = [C_1 \ C_1 \ C_2 \ C_2] \quad (10.9)$$

Thus,

$$G_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Similarly, for $N = 8$, G_8 can be constructed from G_4 .

If G_4 is represented as

$$G_4 = [C_1 \ C_2 \ C_3 \ C_4] \quad (10.10)$$

where C_1, C_2, C_3 and 4 stand for columns 1, 2, 3, and 4 of G_4 .

Then G_{41} is obtained by inserting zero column vectors as

$$G_{41} = [C_1 \ 0 \ C_2 \ 0 \ C_3 \ 0 \ C_4 \ 0] \quad (10.11)$$

G_{42} is obtained by duplicating the columns as

$$G_{42} = [C_1 \ C_1 \ C_2 \ C_2 \ C_3 \ C_3 \ C_4 \ C_4] \quad (10.12)$$

$$G_8 = \begin{bmatrix} G_{41} \\ G_{42} \end{bmatrix} \quad (10.13)$$

Hence,

$$G_8 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Generalizing this, G_N can be formed from G_{N-1} as

$$G_N = \begin{bmatrix} G_{(N/2)1} \\ G_{(N/2)2} \end{bmatrix} \quad (10.14)$$

B. Generator matrix construction with information vector permuted

If the information input vector is permuted, then the generator matrix G_N is given by

$$G_N = F^{\otimes n} \quad \{N = 2^n, n \geq 0\} \quad (10.15)$$

For $N = 4$, generator matrix G_4 is given by

$$G_4 = F^{\otimes 2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Similarly, for $N = 8$, generator matrix G_8 is given by

$$G_8 = F^{\otimes 3} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

10.1.2 Channel Splitting Phase

After obtaining the vector channel W_N from the channel combining phase by using N copies of W , W_N is to be splitted back into a set of N binary input coordinate channels W_N^i , $1 \leq i \leq N$.

Define a channel $W_N^1 : u_1 \rightarrow y_1, y_2, \dots, y_N$
with input u_1 and outputs

Define a channel $W_N^2 : u_2 \rightarrow y_1, y_2, \dots, y_N, \hat{u}_1$

Define a channel $W_N^3 : u_3 \rightarrow y_1, y_2, \dots, y_N, \hat{u}_1, \hat{u}_2$

Generalizing this, define a channel

Define a channel $W_N^i : u_i \rightarrow y_1, y_2, \dots, y_N, \hat{u}_1, \hat{u}_2, \dots, \hat{u}_{i-1}$

For $N = 4$, the channel splitting and the split channel W_4^3 are shown in Fig. 10.6.

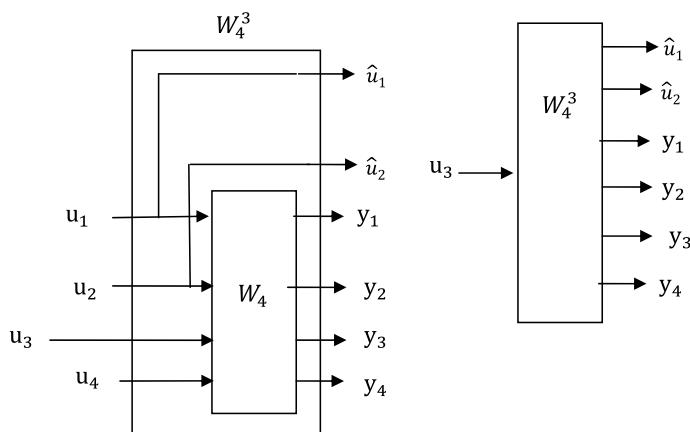
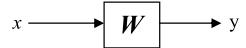


Fig. 10.6 Channel splitting and splitting channel W_4^3

Fig. 10.7 Binary erasure channel



10.1.3 Polarization of Binary Erasure Channels

Consider the Binary Erasure Channel (BEC) shown in Fig. 10.7.

If the Bhattacharya parameter of W is denoted by $Z(W) = \varepsilon$, where ε is the erasure probability of BEC. Denoting W by W_1^1 , then for $N = 2$, the Bhattacharya parameter of the split channel W_2^1 is given by

$$Z(W_2^1) = 2\varepsilon - \varepsilon^2 \quad (10.16)$$

and the Bhattacharya parameter of the split channel W_2^2 is given by

$$Z(W_2^2) = \varepsilon^2 \quad (10.17)$$

The Bhattacharya parameters $Z(W_2^1)$ and $Z(W_2^2)$ can be expressed in terms of $Z(W_1^1)$ as follows

$$Z(W_2^1) = 2Z(W_1^1) - (Z(W_1^1))^2 \quad (10.18a)$$

$$Z(W_2^2) = (Z(W_1^1))^2 \quad (10.18b)$$

Which can be generalized for binary split binary erasure channels for any N value as [2]

$$Z(W_N^{2i-1}) = 2Z(W_{N/2}^i) - (Z(W_{N/2}^i))^2 \quad (10.19a)$$

$$Z(W_N^{2i}) = (Z(W_{N/2}^i))^2 \quad (10.19b)$$

where $Z(W_1^1) = \varepsilon$.

The capacity and the Bhattacharya parameter are related oppositely to each other, i.e., $C(W_N^i) = 1 - Z(W_N^i)$.

Hence, Eqs. (10.18a) and (10.18b) imply that

$$C(W_N^{2i}) = 2C(W_{N/2}^i) - (C(W_{N/2}^i))^2 \quad (10.20a)$$

$$C(W_N^{2i-1}) = (C(W_{N/2}^i))^2 \quad (10.20b)$$

where $C(W_1^1) = 1 - \varepsilon$.

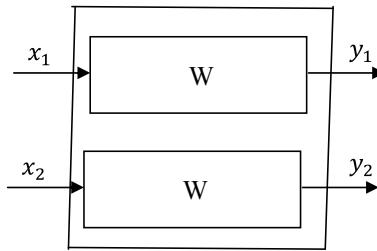


Fig. 10.8 Compound channel with two independent uses of BEC

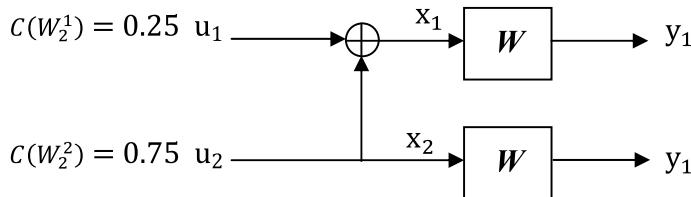


Fig. 10.9 Two polarized BECs, $N = 2^1$

The channel polarization can be illustrated in BEC with erasure probability 0.5 as follows:

The erasure probability of this BEC is 0.5. Hence, the capacity of this channel $C(W) = 1 - 0.5 = 0.5$.

A compound channel can be formed by combining two independent uses of BEC as shown in Fig. 10.8. The compound channel capacity is $2C(W)$.

An equivalent compound channel can be obtained by applying modulo-2 addition operation between the two independent BECs still having the capacity $2C(W)$. The equivalent compound channel is shown in Fig. 10.9.

By using the chain rule of mutual information, the compound channel can be decomposed into two synthesized channels W_2^1 (bad channel) with input u_1 and the y_1 and y_2 as outputs and W_2^2 (good channel) with input u_2 and u_1 , y_1 and y_2 as outputs. The two independent BECs with the same reliability are transformed into two polarized channels after channel combining and channel splitting. The sum capacity of two channels is unchanged, that is $C(W_2^1) + C(W_2^2) = 2C(W)$.

The two polarized BECs are shown in Fig. 10.9. For $N = 2^1 = 2$, $i = 1$.

Using Eqs. (10.19a) and (10.19b), the capacities of the two polarized channels are

$$C(W_2^1) = (0.5)^2 = 0.25; \quad C(W_2^2) = 1 - (0.5)^2 = 0.75$$

Among the polarized channels, good channel has larger capacity.

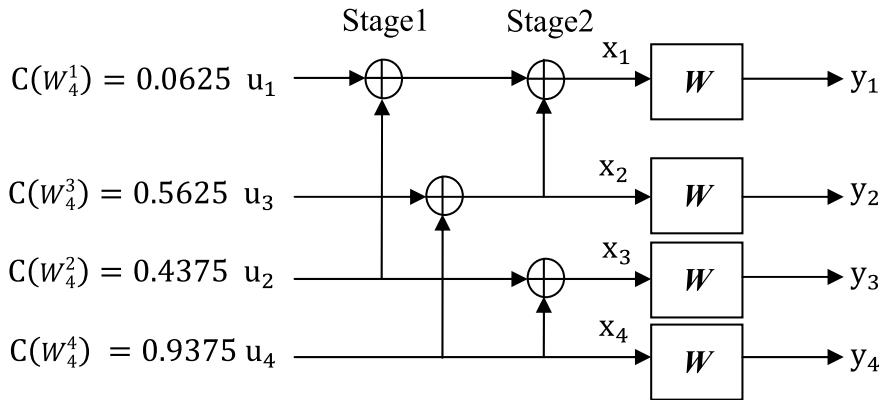


Fig. 10.10 Four polarized BECs, $N = 2^2$

For four independent uses of the given BEC, after channel combining and channel splitting, the four independent channels can be decomposed into four polarized channels W_4^1, W_4^2, W_4^3 and W_4^4 as shown in Fig. 10.10. For $N = 4 = 2^2, i = 2$.

Using Eqs. (10.19a) and (10.19b), the capacity of the channel W_4^1 is given by $W_4^1 = (W_2^1)^2 = 0.0625$, the capacity of the channel W_4^2 is given by $W_4^2 = 2C(W_2^1) - (C(W_2^1))^2 = 0.5 - 0.0625 = 0.4375$, the capacity of the channel W_4^3 is evaluated as $W_4^3 = (C(W_2^2))^2 = 0.5625$ and the capacity of the channel W_4^4 is evaluated as $W_4^4 = 2C(W_2^2) - (C(W_2^2))^2 = 1.5 - 0.5625 = 0.9375$. One can observe that the capacity of channel W_4^1 is reduced as compared to the capacity of W_2^1 and the capacity of W_4^4 is increased as compared to the capacity of W_2^2 . Thus, the polarization effect with four channels is more significant.

The following MATLAB Program 10.1 can be used for calculating the capacities of the polarized channels for N .

Program 10.1

```

clear all;clc;
i=1:3;
N=2.^i;
C(1,1)=0.5;
for i=1:1:3
WN=N(1,i);
for j=1:WN/2
C(WN,2*j-1)=C(WN/2,j)^2;
C(WN,2*j)=2*C(WN/2,j)-C(WN/2,j)^2;
end
end
WN=8;
C(WN,1:WN)

```

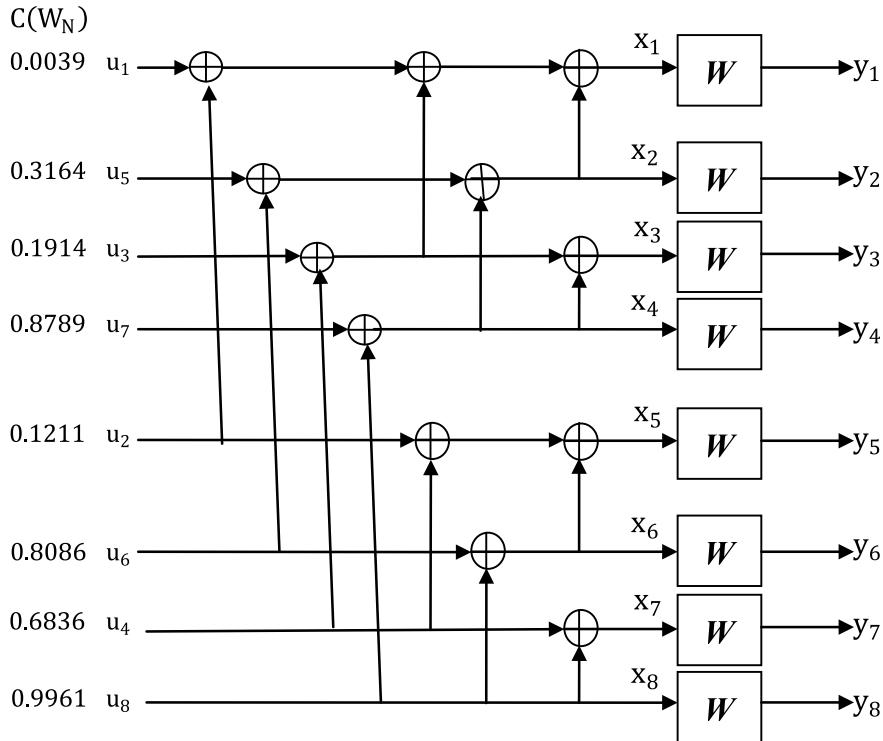


Fig. 10.11 Eight polarized BECs, $N = 2^3$

The execution of the above program for $N = 8$ and $i = 3$ yields the following capacities of the polarized channels

$$0.0039 \ 0.1211 \ 0.1914 \ 0.6836 \ 0.3164 \ 0.8086 \ 0.8789 \ 0.9961$$

For eight independent uses of the given BEC, after channel combining and channel splitting, the eight independent channels can be decomposed into eight polarized channels as shown in Fig. 10.11.

The polarization effect can be performed by use of $N = 2^n$ independent BECs, and the capacities of the polarized channels are evaluated recursively. If the code length N approaches infinity, the polarized channels capacities go to two extremes 0 or 1.

10.1.4 Polarization Theorem

The bit-channel capacities $\{C(W_i)\}$ polarize: for any $\delta \in (0, 1)$, as the construction size N grows

$$\left[\frac{\text{no. channels with } C(W_i) > 1 - \delta}{N} \right] \rightarrow C(W)$$

and

$$\left[\frac{\text{no. channels with } C(W_i) < \delta}{N} \right] \rightarrow 1 - C(W)$$

where $\delta \approx 2^{-\sqrt{N}}$.

From the theorem, it can be observed that the channel capacities tend to move toward 1 or 0. The conditional entropy lies between δ and $1 - \delta$.

10.1.5 Polarization of AWGN Channels

The polarization effect can be generalized for any discrete memoryless channels and channels with continuous outputs such as AWGN channel.

Equations (10.19a) and (10.19b) still can be used to determine the split channel capacities for AWGN channels also. However, the initial value should be chosen as

$$Z(W_1^1) = e^{-\frac{RE_b}{N_0}} \quad (10.21)$$

where R is the code rate.

For the AWGN channel, the polarization effect applies not only to the capacity but also to the bit error rate (BER). That is, the BER tends to zero for the good subchannels and 0.5 for the bad ones

10.2 Polar Encoder Structures and Encoding

The two polar encoder structures are

1. Polar encoder with unpermuted input
2. Polar encoder with permuted input

The two polar encoding schemes are:

1. Non-systematic encoding
2. Systematic encoding

For given code length N , information bits of length K , the binary input vector $u = (u_1, u_2, \dots, u_N)$ can be formed with K information bits and $N - K$ frozen bits. Then the polar code word x with code rate $R = K/N$ is given by

$$x = uG_N \quad (10.22)$$

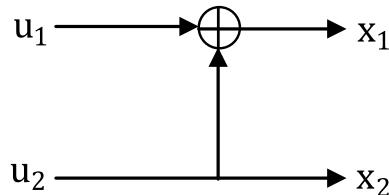


Fig. 10.12 Polar encoder for $N = 2$

In non-systematic polar code, the information bits do not appear as a part of encoded output.

10.2.1 Polar Encoder for $N = 2$

Polar encoder for $N = 2$ is shown in Fig. 10.12

$$\begin{aligned}x_1 &= u_1 \oplus u_2 \\x_2 &= u_2\end{aligned}$$

10.2.2 Polar Encoder for $N = 4$

Polar encoder for $N = 4$ is shown in Fig. 10.13.

$$\begin{aligned}x_1 &= u_1 \oplus u_2 \oplus u_3 \oplus u_4 \\x_2 &= u_3 \oplus u_4 \\x_3 &= u_2 \oplus u_4 \\x_4 &= u_4\end{aligned}$$

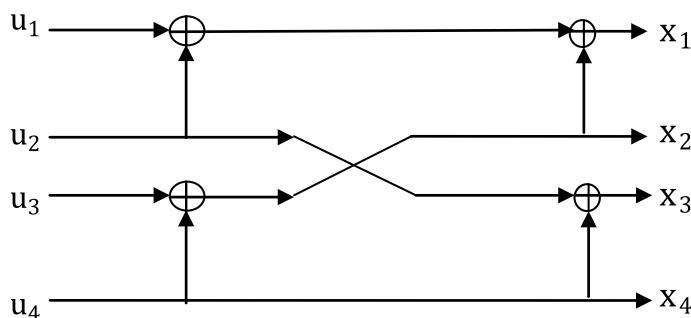
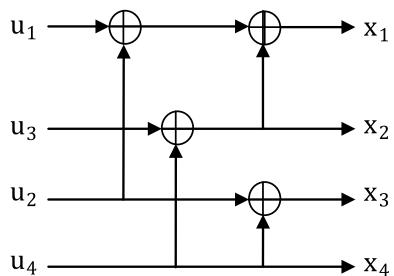


Fig. 10.13 Polar encoder for $N = 4$

Fig. 10.14 Polar encoder for $N = 4$ with permuted input data vector



10.2.3 Polar Encoder for $N = 4$ with Input Data Vector Permuted

Polar encoder shown in Fig. 10.13 can be redrawn as shown in Fig. 10.14 with the input data vector permuted.

10.2.4 Polar Encoder for $N = 8$

Polar encoder for $N = 8$ is shown in Fig. 10.15.

$$\begin{aligned}x_1 &= u_1 \oplus u_2 \oplus u_3 \oplus u_4 \oplus u_5 \oplus u_6 \oplus u_7 \oplus u_8 \\x_2 &= u_5 \oplus u_6 \oplus u_7 \oplus u_8 \\x_3 &= u_3 \oplus u_4 \oplus u_7 \oplus u_8\end{aligned}$$

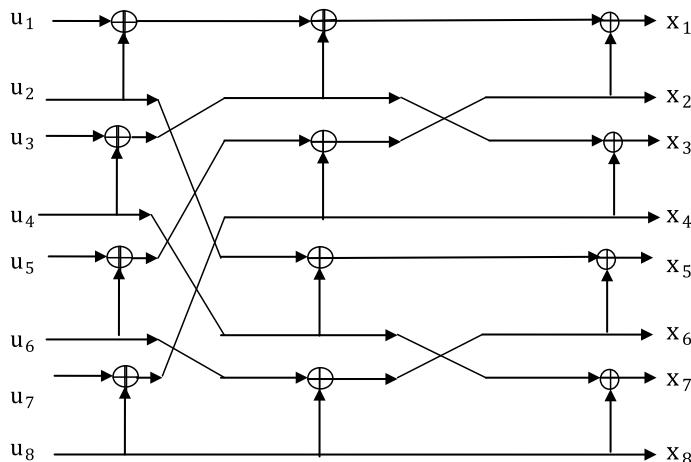


Fig. 10.15 Polar encoder for $N = 8$

$$\begin{aligned}x_4 &= u_7 \oplus u_8 \\x_5 &= u_2 \oplus u_4 \oplus u_6 \oplus u_8 \\x_6 &= u_6 \oplus u_8 \\x_7 &= u_4 \oplus u_8 \\x_8 &= u_8\end{aligned}$$

Example 10.1 Find the code word for input data vector $u = [1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1]$ using the encoder structure shown in Fig. 10.15 for $N = 8$.

Solution

$$u = [1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1]$$

The code vector

$$X = uG_8$$

$$\begin{aligned}&= [1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1] \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \\&= [1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1]\end{aligned}$$

10.2.5 Polar Encoder for N = 8 with Input Data Vector Permuted

Polar encoder shown in Fig. 10.15 can be redrawn as shown in Fig. 10.16 with the input data vector permuted.

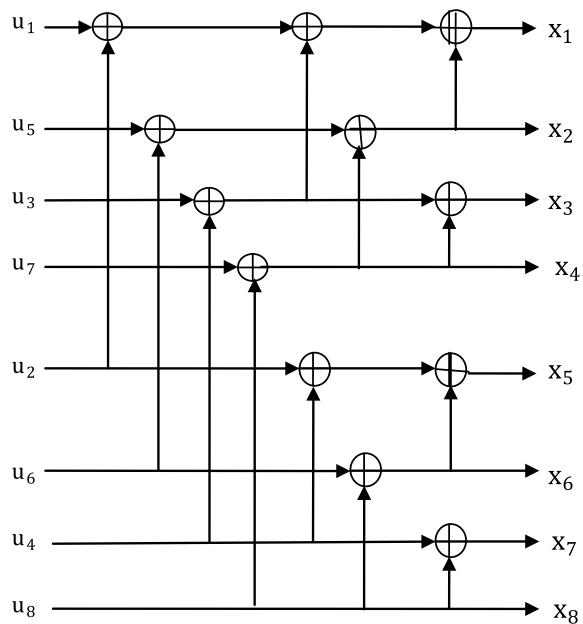
It may be preferable to use the structure shown in Fig. 10.16 in order to simplify the implementation.

Example 10.2 Find the code word for input data vector of Example 10.1 using the encoder structure shown in Fig. 10.16 for $N = 8$.

Solution

$$u = [1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1]$$

Fig. 10.16 Polar encoder for $N = 8$ with permuted input data vector



The permuted input vector is

$$[u_1 \ u_5 \ u_3 \ u_7 \ u_2 \ u_6 \ u_4 \ u_8] = [1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1]$$

The code vector

$$X = uG_8 = uF^{\otimes 3}$$

$$= [1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1] \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} = [1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1]$$

10.2.6 Non-systematic Polar Encoding Using MATLAB

The following MATLAB Program 10.2 is written to find the code word consider a polar encoder with $N = 8$, rate = 1/2 and information bits $[1 \ 0 \ 1 \ 1]$.

Program 10.2 MATLAB program to compute codeword with $N = 8$, rate = 1/2

```

clc; clear all;
n=3;K=4;N=2^n;
e=0.5;%erasure probability
curN=2.^[1:n];
Z(1,1)=e;
    for i = 1 : N/2
        for ind=1:n
            Z(curN(ind),2*i-1)=2*Z(curN(ind)/2,i)-
            (Z(curN(ind)/2,i))^2;
            Z(curN(ind),2*i)=(Z(curN(ind)/2,i))^2;
        end
    end
    [sortedZ,Cind]=sort(Z(N,:));
    cindex=sort(Cind(1:K));
    u=zeros(1,N);
    inf=[1 0 1 1];%information bits
    %allotion of information bits to good channels
    for i=1:K
        u(cindex(i))=inf(i);
    end
    index=[1 5 3 7 2 6 4 8];
    for i=1:8
        up(i) =u(index(i));% permuted input vector
    end
    %Generation of G matrix
    F = [1,0;1,1];
    FN = zeros(N,N);
    FN(1:2,1:2) = F;
    for i = 2:log2(N)
        FN(1:2^i,1:2^i) = kron(FN(1:2^(i-1),1:2^(i-1)),F);
    end
    x=mod(up*FN,2); %out put of encoder

```

The execution of the above program yields $u_4 \ u_6 \ u_7 \ u_8$ as the good channels and assigning the information bits to good channels and zeros to other frozen channels, input vector to the encoder is formed. The program has yielded the code word, i.e., output of the encoder as $[1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1]$.

10.2.7 Systematic Polar Encoding

In a systematic polar code, the message vectors are mapped to code words, such that the message bits are explicitly visible in the code word.

A rate 1/2 systematic polar encoder with information bits $u_4 u_6 u_7 u_8$ and frozen bits $u_1 u_2 u_3 u_5$ as zeros is shown in Fig. 10.17.

The encoder's input goes into a pre-processor, which solves for the unique information bits that cause the encoder output to have the information bits $u_4 u_6 u_7 u_8$ as shown in Fig. 10.17.

The systematic polar encoding of an information of K bits is the solution of

$$x = u_{pp} \cdot F^{\otimes n} \quad (10.23)$$

where

$$\begin{aligned} x &= [x_1 \ x_2 \ x_3 \ u_4 \ x_5 \ u_6 \ u_7 \ u_8] \\ u_{pp} &= [0 \ 0 \ 0 \ u_{pp7} \ 0 \ u_{pp6} \ u_{pp4} \ u_{pp8}] \end{aligned}$$

Example 10.3 Find a code for systematic encoder for $N = 8$ with information bits $\{u_4, u_6, u_7, u_8\} = \{1, 0, 1, 1\}$.

Solution For $\{u_4, u_6, u_7, u_8\} = \{1, 0, 1, 1\}$, Eq. (2.23) becomes

$$[x_1, x_2, x_3, 1, x_5, 0, 1, 1]$$

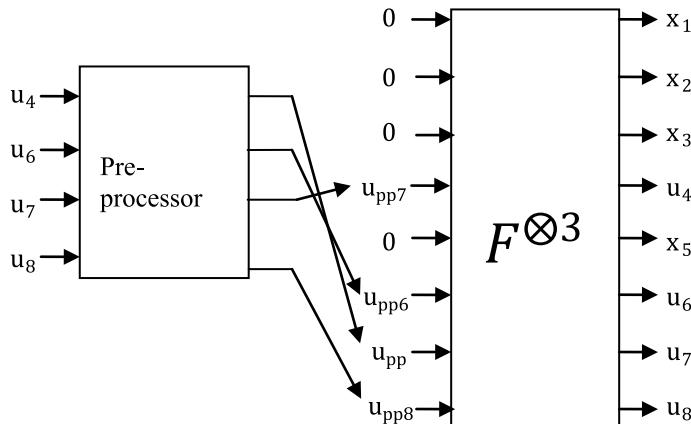


Fig. 10.17 Systematic polar encoder for $N = 8$, code rate = 1/2

$$= [0, 0, 0, u_{pp7}, 0, u_{pp6}, u_{pp4}, u_{pp8}] \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad (10.24)$$

The above relation leads to the following equations

$$\begin{aligned} u_{pp4} + u_{pp6} + u_{pp7} + u_{pp8} &= x_1 \\ u_{pp6} + u_{pp7} + u_{pp8} &= x_2 \\ u_{pp4} + u_{pp7} + u_{pp8} &= x_3 \\ u_{pp7} + u_{pp8} &= 1 \\ u_{pp4} + u_{pp6} + u_{pp8} &= x_5 \\ u_{pp6} + u_{pp8} &= 0 \\ u_{pp4} + u_{pp8} &= 1 \\ u_{pp8} &= 1 \end{aligned}$$

Solving the above equations, we get

$$u_{pp4} = 0; \quad u_{pp7} = 0; \quad u_{pp6} = 1$$

Using the values of u_{pp4} , u_{pp6} , u_{pp7} and u_{pp8} , we get

$$\begin{aligned} x_5 &= 1 + 0 + 1 = 0 \\ x_3 &= 0 + 0 + 1 = 1 \\ x_2 &= 0 + 1 + 1 = 0 \\ x_1 &= 0 + 1 + 0 + 1 = 0 \end{aligned}$$

The code $X = [0, 0, 1, 1, 0, 0, 1, 1]$.

Gaussian elimination method requiring the number of XORs approximately equal to the number of ones in the matrix can be used to solve the equations as $F^{\otimes n}$ is an upper triangular matrix. The complexity is much more than $(N \log N)$ XORs as indicated by Arikan.

10.2.8 Efficient Systematic Polar Encoding Algorithm

An efficient algorithm to solve Eq. (10.23) is reported in [3]. This algorithm requires the least number of XORs, exactly equal to that of a non-systematic polar encoder.

The following MATLAB Program 10.2 is written to find the code word using considering a systematic polar encoder with $N = 8$, rate = $1/2$, and information bits $[1\ 0\ 1\ 1]$.

The execution of the MATLAB program has yielded $[0\ 0\ 1\ 1\ 0\ 0\ 1\ 1]$ as the output of the encoder in which the information bits are $[1\ 0\ 1\ 1]$.

Program 10.3 MATLAB program to compute codeword with $N = 8$, rate = $1/2$, and erasure probability -0.5 for systematic polar encoder

```
clear all;clc;
N=8;
n=3;
K=4;
design_epsilon=0.5
IWi = zeros(N,1);
IWi(1) = 1 - design_epsilon;
I_tmp = zeros(N,1);
m = 1;
while(m <= N/2)
    for k = 1 : m
        I_tmp(k) = IWi(k) * IWi(k);
        I_tmp(k+m) = 2 * IWi(k) - IWi(k) * IWi(k);
    end
    IWi = I_tmp;
    m = m * 2;
end
[~, sorted_indices] = sort(IWi);
y=zeros(1,N);
inf=[1 0 1 1];
x=y;
sortindicesinf=sorted_indices(N-K+1:end) ;
sorted_indicesFz=zeros(8,1);
sorted_indicesFz(sorted_indicesinf)=-1;
x(sorted_indicesFz== -1)=inf;
X = zeros(N,1+n); %initialize
X(:,1) = y;
X(:,n+1) = x;
```

```

for i=N:-1:1
    if(sorted_indicesFz(i)==-1)
        s=n+1;
        delta=-1;
    else
        s=1;
        delta=1;
    end
    str=dec2bin(i-1,n);
    for j=1:n
        t = s+j*delta;
        l=min([t,t-delta]);
        kappa=2^(n-1);
        if(str(l)=='1')
            X(i,t) = X(i,t-delta);
        else
            X(i,t) = mod(X(i,t-delta)+X(i+kappa,t-
delta), 2);
        end
    end
end
y=X(:,1);
x=X(:,n+1);

```

10.3 Polar Decoding

10.3.1 Successive Cancellation Decoding

The successive cancelation decoding (SCD) algorithm for polar codes decoding has been proposed by Arikan [1]. The butterfly diagram for SCD is same as that of the encoder butterfly diagram. In the decoding butterfly diagram, the likelihood ratios (LRs) propagate from right to left. The nodes at upper corner and lower corner of the butterfly are called the upper node and lower node, respectively. The proper decoding is performed by computing LRs at each node.

The likelihoods evolve in the reverse direction from right to left. The likelihood ratio is defined as

$$L_N^{(i)}(y_1^N, u_1^{i-1}) = \frac{W_N^{(i)}(y_1^N, u_1^{i-1} | u_i = 0)}{W_N^{(i)}(y_1^N, u_1^{i-1} | u_i = 1)} \quad (10.25)$$

$$\hat{u}_i = \begin{cases} 0, & \text{if } L_N^{(i)}(y_1^N, u_1^{i-1}) \geq 1 \\ 1, & \text{otherwise} \end{cases} \quad (10.26)$$

Polar coding usually comes with SC decoder as a suboptimal decoding method. Nowadays, even though there are some decoding methods performing better, SC decoding still plays an integral role in polar coding. The decoder should know the knowledge of u_1^{i-1} when decoding u_i . Generally, the decoder only knows the values of frozen bits $\{u_j, j \in A^c\}$ from the frozen set A^c in advance. As the SC decoder decodes bits consecutively, at least provides an estimate of u_1^{i-1} when decoding u_i .

Consider a polar code with the parameter vector (N, K, A, u_{A^c}) , where N is the code length, K stands for number of information bits, A is the information set, and u_{A^c} is the frozen bits vector.

The decoder needs to retrieve the information and generate an estimate \hat{u}_1^N of u_1^N with knowledge of y_1^N, A and u_{A^c} . The likelihood ratio (LR) is defined as

$$L_N^{(i)}(y_1^N, u_1^{i-1}) = \frac{W_N^{(i)}(y_1^N, u_1^{i-1} | u_i = 0)}{W_N^{(i)}(y_1^N, u_1^{i-1} | u_i = 1)} \quad (10.27)$$

For convenience, the frozen bits are always set to 0. Then a SC decoder would take the following steps to achieve translating the source bits. For each i from 1 to N :

- If $i \in A^c$, $\hat{u}_i = u_i$.
- If $i \in A$, calculate the LR $L_N^{(i)}(y_1^N, u_1^{i-1})$ and make the decision as

$$\hat{u}_i = \begin{cases} 0, & \text{if } L_N^{(i)}(y_1^N, u_1^{i-1}) \geq 1 \\ 1, & \text{otherwise} \end{cases}$$

The LR can be recursively calculated as

$$L_N^{(2i-1)}(y_1^N, \hat{u}_1^{2i-2}) = \frac{L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,0}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}) L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}) + 1}{L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,0}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}) + L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2})} \quad (10.28)$$

$$L_N^{(2i)}(y_1^N, \hat{u}_1^{2i-1}) = \left[L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,0}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}) \right]^{1-2\hat{u}_{2i-1}} L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}) \quad (10.29)$$

The recursive calculation of LRs can be traced back to code length 1 with the LR $L_1^{(1)}(y_i) = W(y_i|0)/W(y_i|1)$. $L_1^{(1)}(y_i)$ is the soft information observed from the channel.

Figure 10.18 shows the implementation of an SC decoder with code length $N = 8$. The calculation of $L_N^{(i)}$ is conducted from right to left. In this figure, f and g represent two functions as follows

$$f(a, b) = \frac{1 + ab}{a + b} \quad (10.30)$$

$$g(a, b, \hat{u}_{\text{sum}}) = a^{1-2\hat{u}_{\text{sum}}} b \quad (10.31)$$

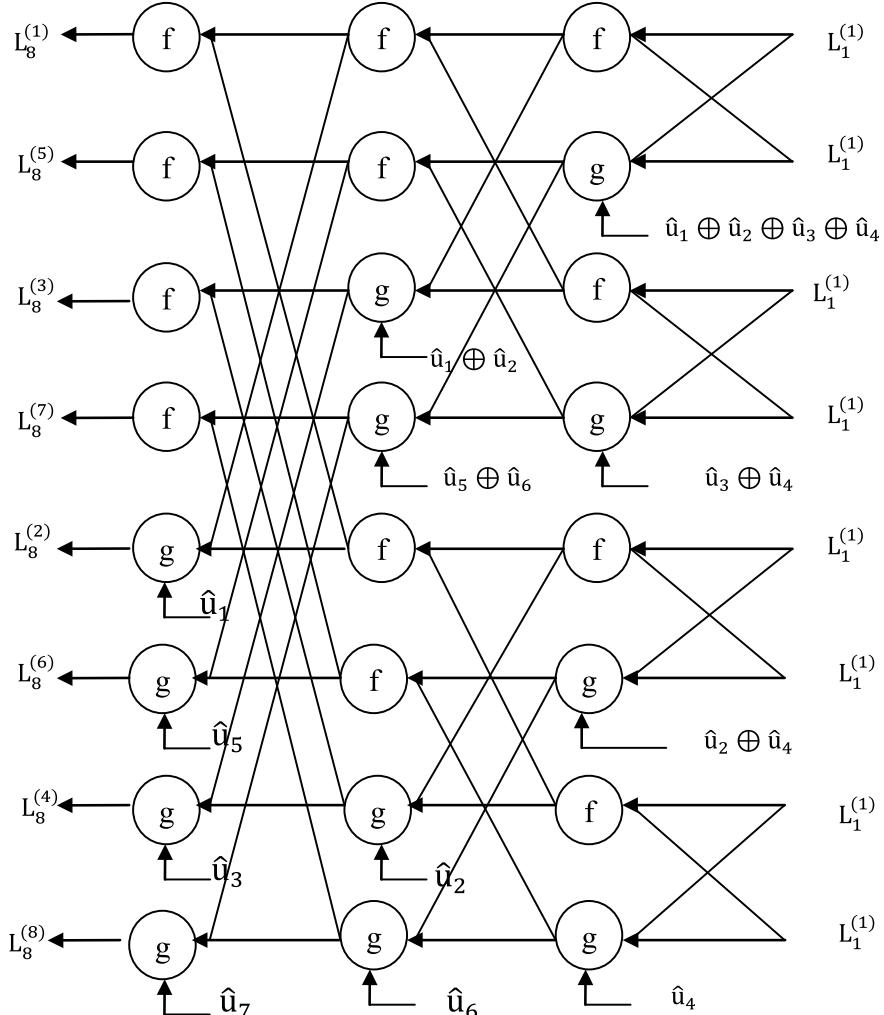


Fig. 10.18 Polar SC decoder for $N = 8$ using SCD

The variables a and b denote $L_{N/2}^{(i)}\left(y_1^{N/2}, \hat{u}_{1,0}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}\right)$ and $L_{N/2}^{(i)}\left(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}\right)$, respectively.

10.3.2 SC Decoding Algorithm

The steps of SC decoding algorithm are explained as follows:

Step 1: The rightmost side variable in Fig. 4.7 is initialized with channel observations. For $i = 1, 2, \dots, N$, $L_1^{(1)}(y_i) = W(y_i|u_i = 0)/W(y_i|u_i = 1)$.

Step 2: This step decodes the first bit u_1 . If u_1 is a frozen bit, u_1 is set to 0. Otherwise, the likelihood ratios are updated and get obtain the required LR by calculating f and g . If $\text{LR} > 1$, set u_1 to 0. If $\text{LR} < 1$, u_1 is to be set to 1. If $\text{LR} = 1$, u_1 is to be set to 0 or 1 with equal probability.

Step 3: In this step, the second bit u_2 is decoded. The process is like step 2 except that the value \hat{u}_1 is used for computing g functions to obtain LR value. u_2 is determined from the obtained LR value.

Step 4: This step decodes the remaining bits u_3, \dots, u_N in the order by making use of the estimated previous bits values.

Example 10.4 When polar codes with code length $N = 8$, code rate $R = 1/2$ are transmitted over AWGN channel with the information set $A = \{4, 6, 7, 8\}$, frozen bits u_1, u_2, u_3, u_5, u_7 as 0, the channel outputs are observed as $\{0.4800, -2.7848, 0.8946, -0.7669, 0.5275, -2.3709, 1.1148, -0.7829\}$. Find the information bits u_4, u_6, u_7, u_8 using SC decoding algorithm.

Solution Decoding path and decoding tree of u_4 are shown in Fig. 10.19a and b, respectively.

Decoding path and decoding tree of u_6 are shown in Fig. 10.20a and b, respectively.

Decoding path and decoding tree of u_7 are shown in Fig. 10.21a and b, respectively.

Decoding path and decoding tree of u_8 are shown in Fig. 10.22a and b, respectively (Fig. 10.23).

Example 10.5 When polar codes with code length $N = 8$, code rate $R = 1/3$ are transmitted over AWGN channel with the information set $A = \{4, 6, 8\}$, frozen bits u_1, u_2, u_3, u_5, u_7 are as 0, the channel outputs are observed as $\{1.7097, 1.1458, 1.0989, 1.7938, 0.5978, 1.3483, 1.4175, 0.8781\}$. Find the information bits u_4, u_6, u_8 using SC decoding algorithm.

Solution

Step 1: In the first time slot, the rightmost side variables in Fig. 10.24 are initialized as $\{1.7097, 1.1458, 1.0989, 1.7938, 0.5978, 1.3483, 1.4175, 0.8781\}$. These values are all channel observations.

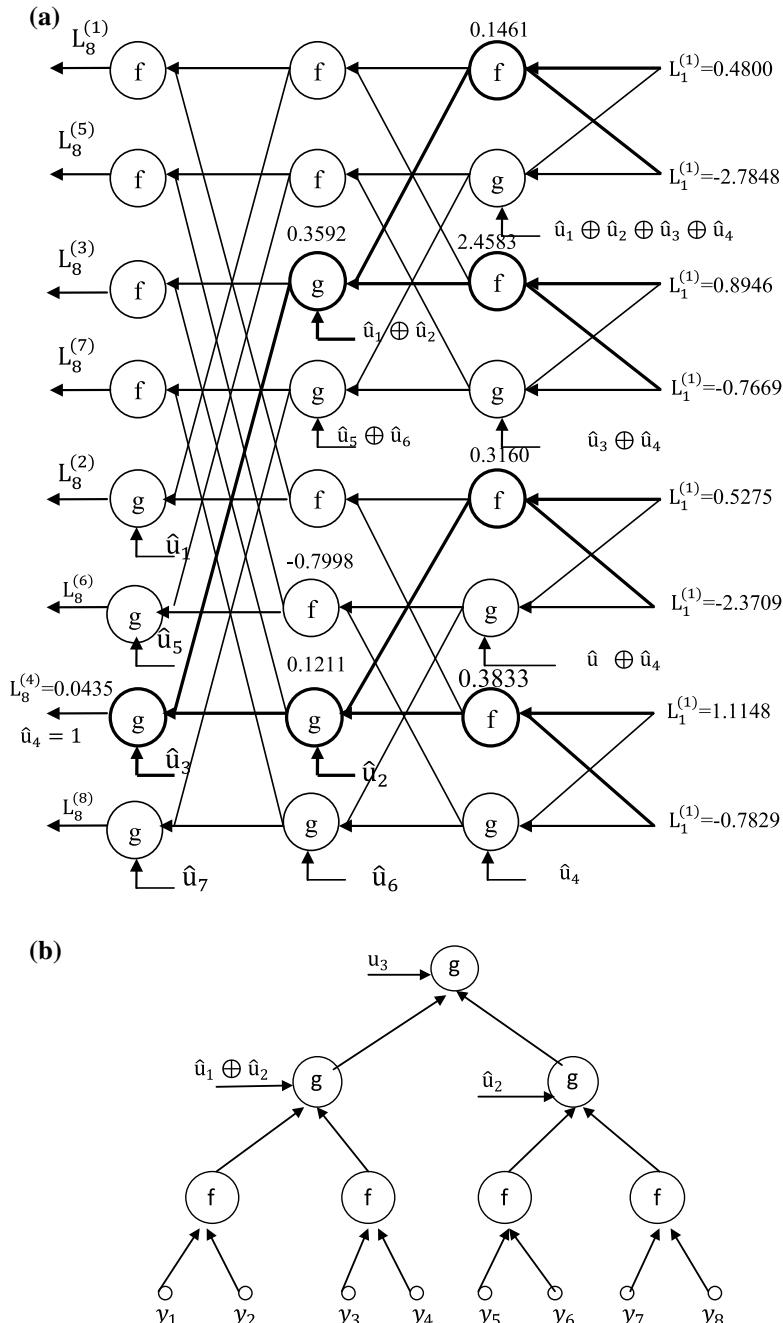


Fig. 10.19 **a** Decoding path of u_4 . **b** Decoding tree of u_4

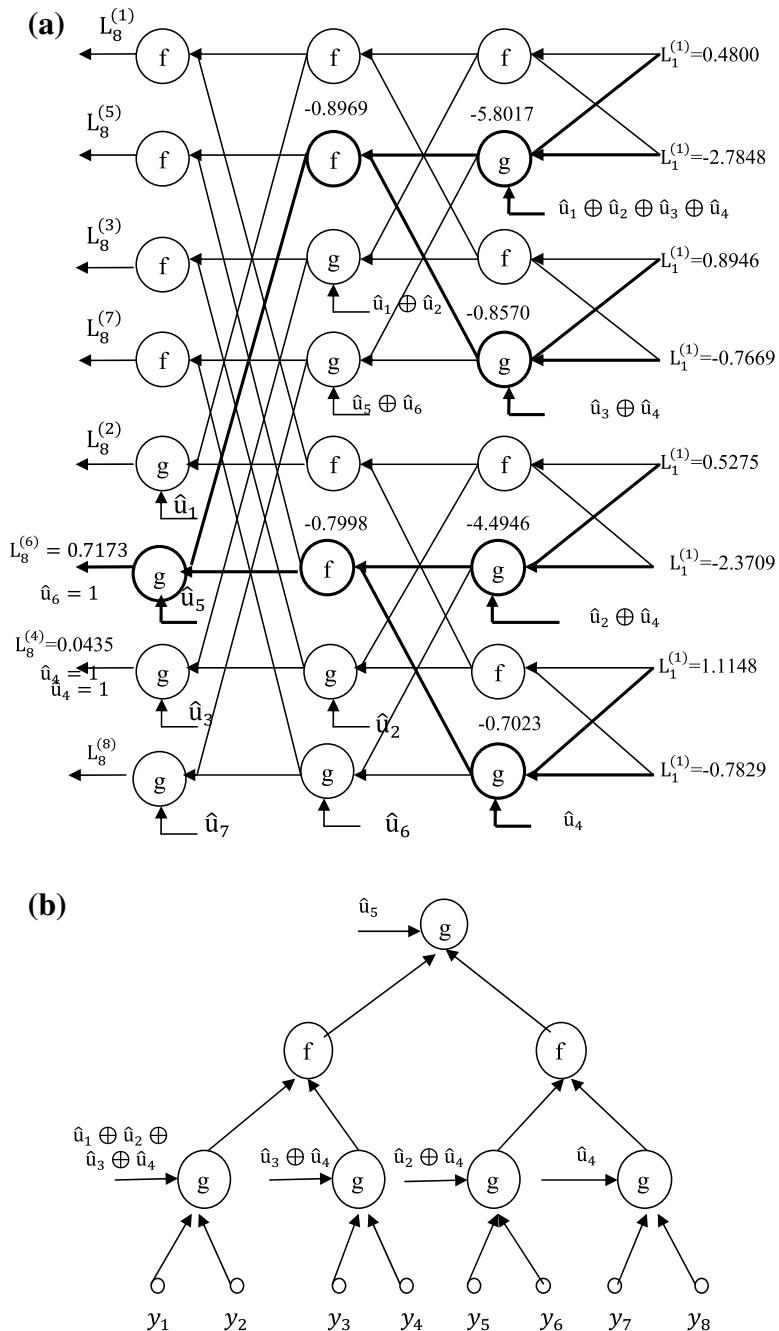


Fig. 10.20 **a** Decoding path of u_6 . **b** Decoding tree of u_6

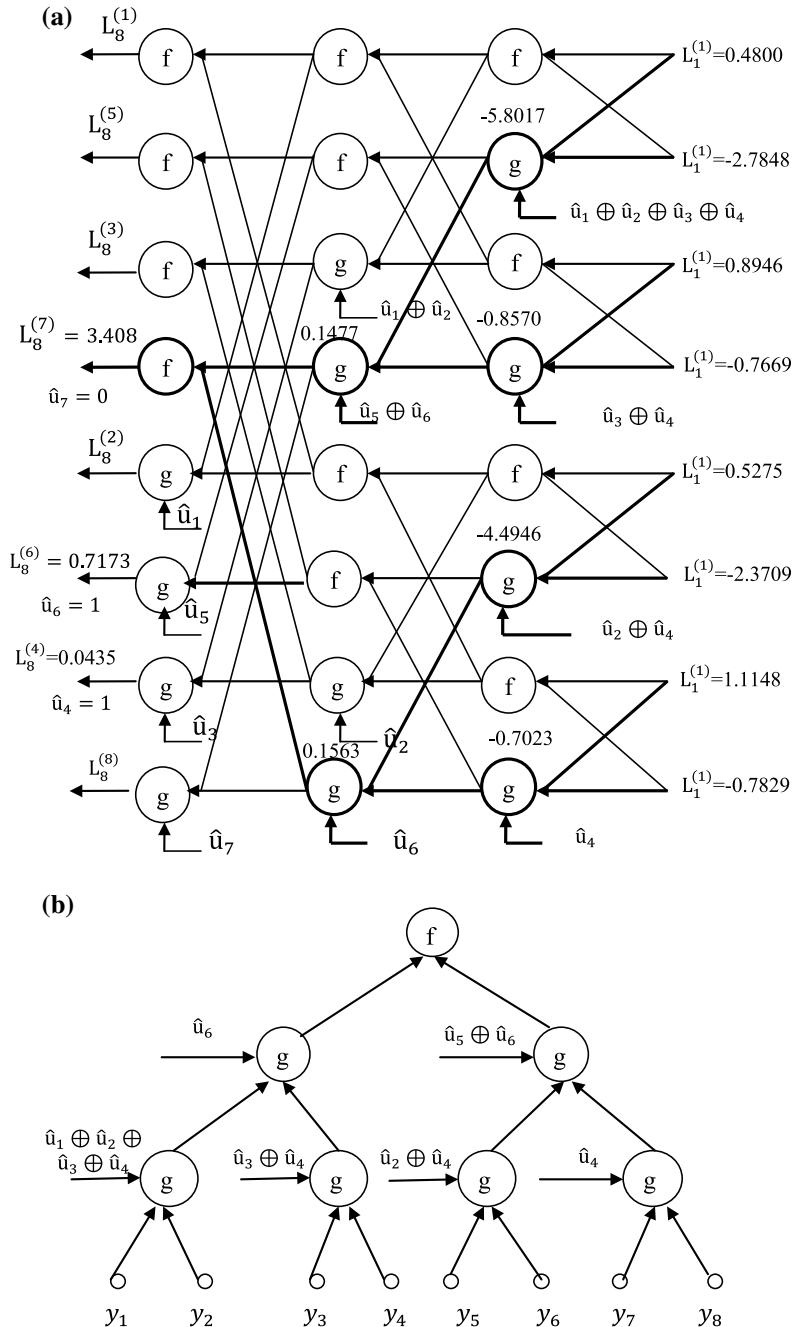


Fig. 10.21 **a** Decoding path of u_7 . **b** Decoding tree of u_7

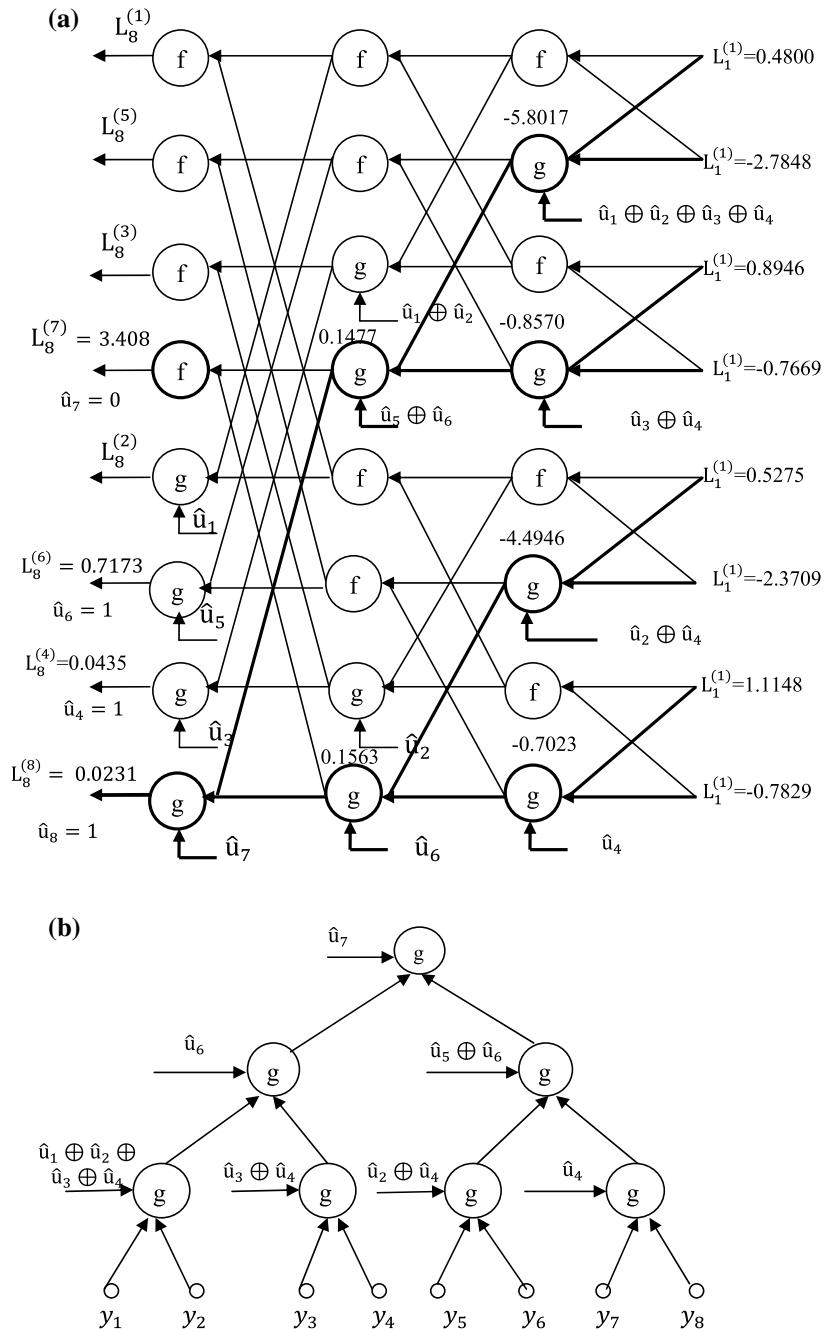


Fig. 10.22 **a** Decoding path of u_8 . **b** Decoding tree of u_8

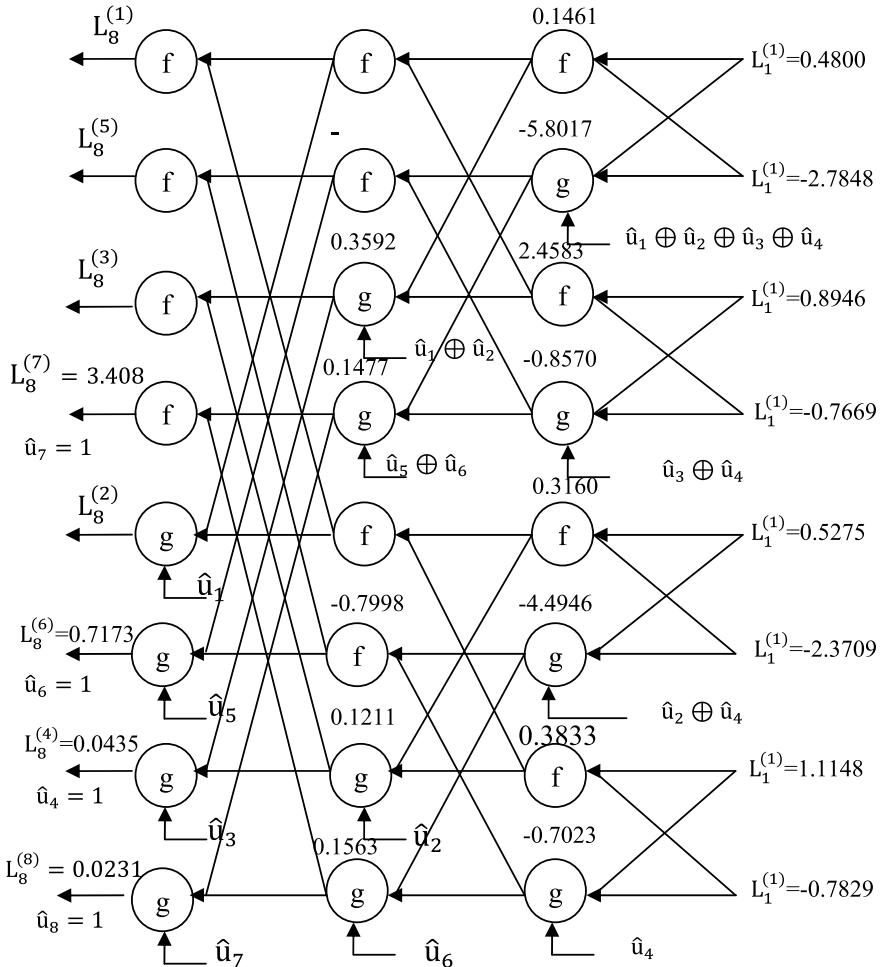


Fig. 10.23 Polar SC decoder for Example 10.4

Step 2: It judges whether u_1 is an information bit before decoding it. Since the frozen set is $\{1, 2, 3, 5\}$, then it knows that u_1 is a frozen bit and its value is 0.

Step 3: Similar to step 2, it is known that u_2 and u_3 are both frozen bits and there is no need to decode them. Skip u_2, u_3 and decode u_4 directly. When decoding u_4 , the likelihood ratio information is updated from right to left. In the second time slot, the rightmost nodes denoted by f function do their LR calculations in parallel. The value 1.0362 is obtained by calculating a f function: $(1 + 1.7097 \times 1.1458)/(1.7097 + 1.1458) = 1.0362$. Similarly, the other functions are calculated and shown in Fig. 10.24 on the top of the f nodes. In the third time slot, the first and third g function nodes of the middle stage of Fig. 10.24 do LR calculations and shown top of the corresponding g nodes. The value 1.0644 is obtained by a g function

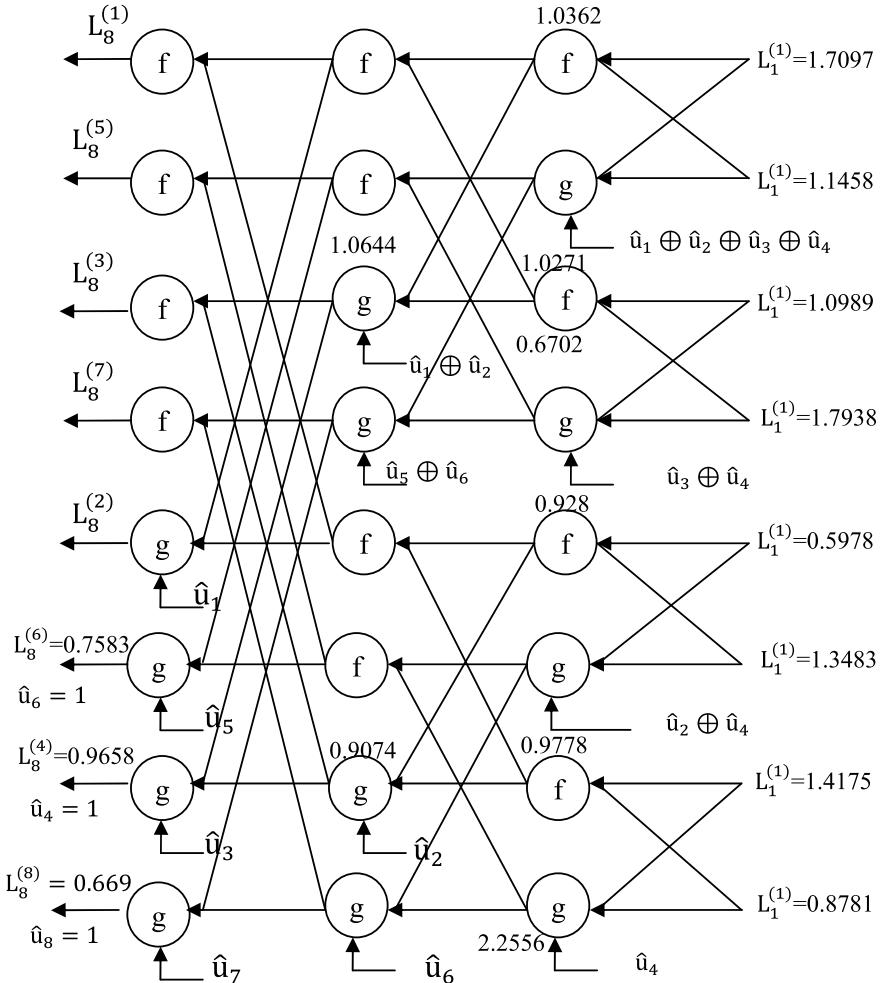


Fig. 10.24 Polar SC decoder for Example 10.5

$1.032^{1-2(\hat{u}_1 \oplus \hat{u}_2)} \times 1.0271 = 1.0644$ since $\hat{u}_1 \oplus \hat{u}_2 = 0$. Similarly, the other value is evaluated as 0.9074. In the fourth time slot, decision on u_4 is made by evaluating the LR $L_8^{(4)}$ by the corresponding g function of the leftmost stage of Fig. 10.24. After calculating $L_8^{(4)} = 1.0644 \times 0.9074 = 0.9658$ since $\hat{u}_3 = 0$. It judges u_4 to be 1 because $L_8^{(4)}$ is smaller than 1.

It is known that u_5 is frozen bit, and there is no need to decode it. Skip u_5 and decode u_6 directly. In the fifth time slot, the rightmost nodes denoted by g function do their LR calculations in parallel and shown on top of the corresponding g function nodes in Fig. 10.24. The value 0.6702 is obtained by a g function $1.7097^{1-2(\hat{u}_1 \oplus \hat{u}_2 \oplus \hat{u}_3 \oplus \hat{u}_4)} \times 1.1458 = 0.6702$ since $\hat{u}_1 \oplus \hat{u}_2 \oplus \hat{u}_3 \oplus \hat{u}_4 = 1$.

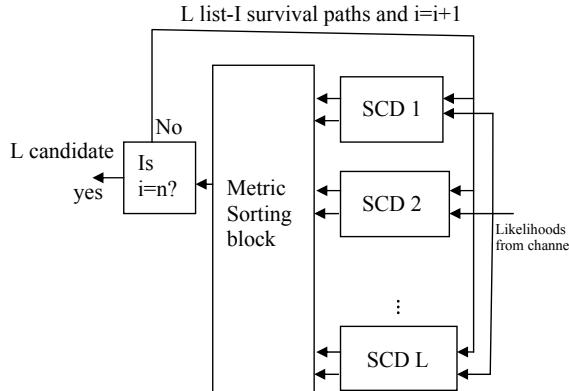


Fig. 10.25 Successive cancellation list decoder

10.3.3 Successive Cancellation List Decoding

The successive cancellation list (SCL) decoding proposed in [4, 5] is proved to provide improved results than SC decoder. In SCL, a list (L) of successive cancellation decoders is used in parallel as shown in Fig. 10.25 to produce a list of candidate decisions and pick the most likely word from the list.

A SCL decoder keeps track of several decoding results instead of one. Since the decoding path is split into two new paths one ending with 0, and the other ending with 1 for each information bit, pruning of the paths is to be done allowing the maximum number of paths to the list size L . The pruning criterion will keep the most likely paths in order to keep the best paths at each stage. The evolution of decoding paths for $L = 4$ with $N = 4$ with all unfrozen bits is shown in Fig. 10.26. In Fig. 10.26(i), the decoding algorithm starts and the first bit can be either 0 or 1. In the second step, in Fig. 10.26(ii), the second bit assumes either 0 or 1 thus the possible words are {00, 01, 10, 11}. However, the number of paths is not greater than $L = 4$. Hence, pruning is not needed. Figure 10.26(iii) shows the all possible options for the first, second, and third bit but the paths are eight. Only four most likely paths are to be tracked. The words {010, 011, 100, 111} are to be kept as shown in Fig. 10.26(iv). The decoding list algorithm continues for the fourth bit in Fig. 10.26(v). But the paths are eight. Therefore, it is pruned to the best $L = 4$ paths in Fig. 10.26(vi). Finally, the decoding algorithm is terminated and the codewords obtained are: {0100, 0110, 0111, 1111}.

10.4 Problems

1. For a binary erasure channel, if the erasure probability is 0.5, calculate the polarized channel capacities for $N = 16$.
2. Construct the polar encoder for $N = 16$.

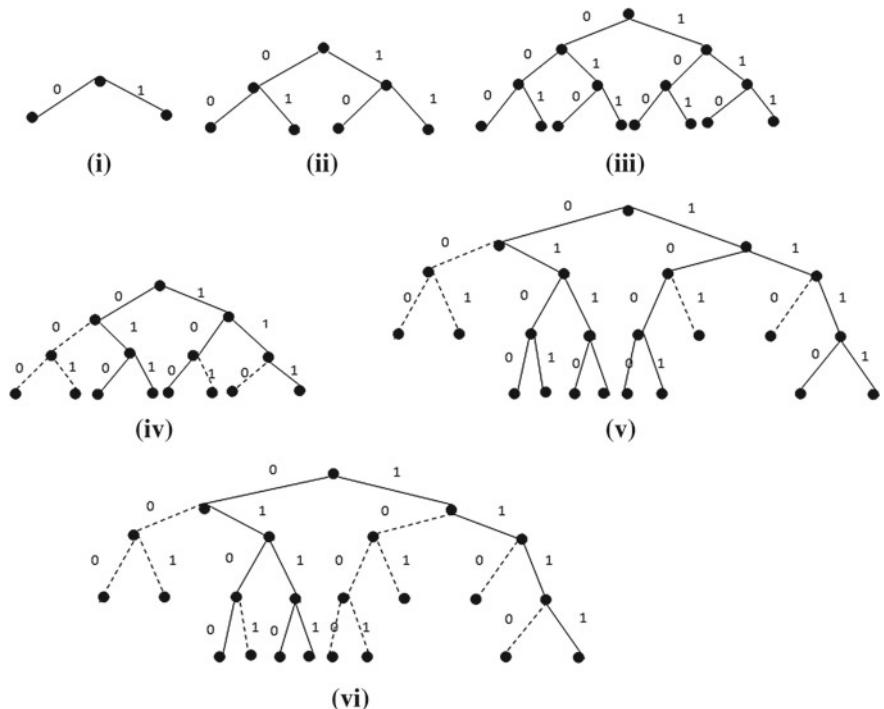


Fig. 10.26 Evolution of decoding paths

3. Construct generator matrix for $N = 16$ from G_8 .
4. Encode the data vector $u = [1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1]$ by a polar encoder and obtain the polar code-word.
5. Polar codes with code length $N = 4$, code rate $R = 1/2$ are transmitted over AWGN channel. The channel outputs are given as $[0.4800 \ -2.7848 \ 0.8946 \ -0.7669]$ and the frozen bits u_1, u_3 are known as 0. Decode the bits u_2 and u_4 .

10.5 MATLAB Exercises

1. Write a MATLAB program for polar decoding using successive cancellation decoding algorithm.
2. Write a MATLAB program for polar decoding using successive cancellation list decoding algorithm.

References

1. E. Arikan, Channel polarization: a method for Constructing capacity achieving codes for symmetric binary-input memory less channels. *IEEE Trans. Inf. Theory* **55**(7), 3051–3073 (2009)
2. O. Gazi, *Polar Codes A Non-Trivial Approach to Channel Coding* (Springer Nature, Singapore, 2019)
3. H. Vangala, Y. Hong, E. Viterbo, Efficient algorithms for systematic polar encoding. *IEEE Commun. Lett.* **20**(1), 1720 (2016)
4. I. Tal, A. Vardy, List decoding of polar codes. *IEEE Trans. Inf. Theory* **61**(5), 2213–2226 (2015)
5. A. Balatsoukas-Stimming, M. Bastani Parizi, A. Burg, LLR based successive cancellation list decoding of polar codes. *IEEE Trans. Signal Process.* **63**(19), 5165–5179 (2015)

Chapter 11

MIMO System



11.1 What Is MIMO?

A channel with multiple antennas at the transmitter and multiple antennas at the receiver is called as a multiple-input multiple-output (MIMO) channel whereas the SISO channel has single antenna at the transmitter and a single antenna at the receiver. A MIMO channel representation is shown in Fig. 11.1.

The key advantages of MIMO system are increased reliability obtained through *diversity* and higher data rate obtained through *spatial multiplexing* [1]. These two concepts are used together in MIMO systems.

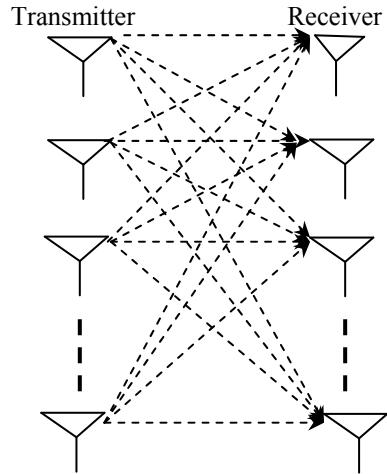
In a diversity system, the same information is transmitted through multiple transmit antennas and received at multiple receive antennas simultaneously. Since the fading for each link between a pair of transmit and receive antennas is considered to be independent and the same information travels through diverse paths and if one path is weak, a copy of information received through the other path may be good and hence, the probability for accurate detection of the information increases.

In a spatial multiplexing, different information can be transmitted simultaneously over multiple antennas, similar to the idea of an OFDM signal thereby boosting the system throughput or capacity of the channel.

11.2 MIMO Channel Model

11.2.1 The Frequency Flat MIMO Channel

Let h_{ji} be a complex number represents the channel gain between i th transmit antenna and j th receive antenna. At a certain time instant, if the symbols $\{s_1, s_2, \dots, s_{N_T}\}$ are transmitted via N_T antennas, then the received signal at antenna j can be expressed as

Fig. 11.1 A MIMO channel

$$y_j = \sum_{i=1}^{N_T} h_{ji} s_i + \eta_j \quad (11.1)$$

With $i = 1, 2, \dots, N_T$ transmitter antennas and $j = 1, 2, \dots, N_R$ receiver antennas, Eq. (11.1) can be represented in matrix form as

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N_R} \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1N_T} \\ h_{21} & h_{22} & \cdots & h_{2N_T} \\ \vdots & \vdots & \ddots & h_{N_R N_T} \\ h_{N_R 1} & h_{N_R 2} & \cdots & \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{N_T} \end{bmatrix} + \begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_{N_R} \end{bmatrix} \quad (11.2)$$

or more compactly as

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \boldsymbol{\eta} \quad (11.3)$$

The fading coefficients H are independent (with respect to both i and j) and identically distributed (i.i.d.). The additive noise at receiver antenna is independent (with respect to η), and identically distributed. It is assumed that the signaling is subject to the average power constraint

$$E[s^2] \leq P \quad (11.4)$$

The H matrix contains the channel coefficients that distort the transmitted signal amplitude and phase in time domain. The channel matrix H is estimated at the receiver and transmitter transmits blindly without any idea of channel information. If the receiver sends back the channel information to the transmitter, then the transmitter is able to adjust the powers allocated to the antennas.

One attractive merit of MIMO systems is the increased antenna diversity, which can alleviate the detrimental effect of flat fading. In a MIMO system with N_T transmit antennas and N_R receive antennas, if the channels for any pair of transmit–receive antennas are independent and experience flat fading, the maximum or full diversity gain is $N_T N_R$. A common way of achieving the full diversity is through space–time (ST) coding, which is discussed in the next chapter.

11.2.2 The Frequency-Selective MIMO Channel

In MIMO systems where any transmit–receive link is subject to multipath fading independently and the channel impulse response is characterized by L resolvable paths, the full diversity gain is $N_T N_R L$ [2, 3]. In frequency-selective MIMO channels, OFDM is usually applied to eliminate the ISI and ICI. To achieve full diversity, coding is used across OFDM subchannels, OFDM blocks, and transmit antennas.

11.2.3 MIMO-OFDM System

In broadband wireless systems, the MIMO channels are severely affected by the frequency-selective fading or potential multipath fading. This fading effect complicates the design of ST codes because of ISI. To overcome this problem, MIMO can be combined with OFDM system, which is referred to as MIMO-OFDM. The combination of MIMO and OFDM has the potential of meeting this stringent requirement since MIMO can improve the capacity and the diversity gain and OFDM can mitigate the detrimental effects due to multipath fading. The schematic block diagram of the MIMO-OFDM system is shown in Fig. 11.2.

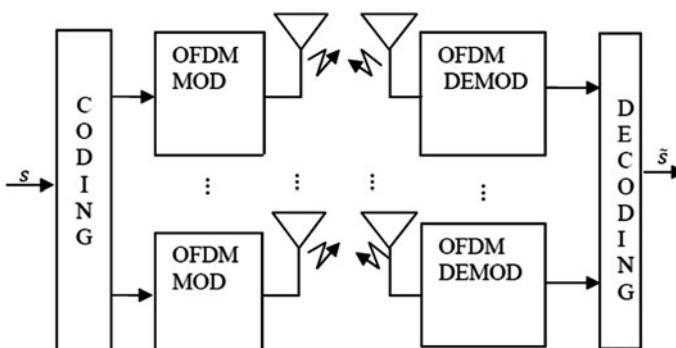


Fig. 11.2 MIMO-OFDM system

The schematic block diagram of MIMO-OFDM system with N_T transmit, N_R receive antennas, and N – tone OFDM is illustrated in Fig. 11.2. The incoming bit stream is first mapped into a number of data symbols by using modulation techniques such as BPSK, QPSK, and QAM. Then, a block of data symbols are encoded into a codeword matrix of size $N_T \times N_T$, and transmitted through N_T transmit antennas in T OFDM blocks, each block having N subchannels. After appending the cyclic prefix on each OFDM block, the blocks will be transmitted through N_T transmit antennas. After passing through the MIMO channels, first the received signals will be sent to the reverse OFDM (cyclic prefix removal, DFT) and then sent to the decoder. If the channel state information (CSI) is available at the receiving side, the optimal ML detection will be performed.

11.3 Channel Estimation

In training-based channel estimation, the used training symbols or pilot tones are known to both the transmitter and the receiver. The knowledge of transmitted pilot symbols at the receiver is exploited to estimate the channel. The block-type pilot arrangement is shown in Fig. 11.3a in which pilot symbols are transmitted periodically for channel estimation. The comb-type pilot arrangement is shown in Fig. 11.3b, where the pilots are transmitted at all times but with an even spacing on the subcarriers for channel estimation.

The estimation can be performed by using LS [4–7]. The training symbols for N subcarriers can be represented by the following diagonal matrix assuming that all subcarriers are orthogonal.

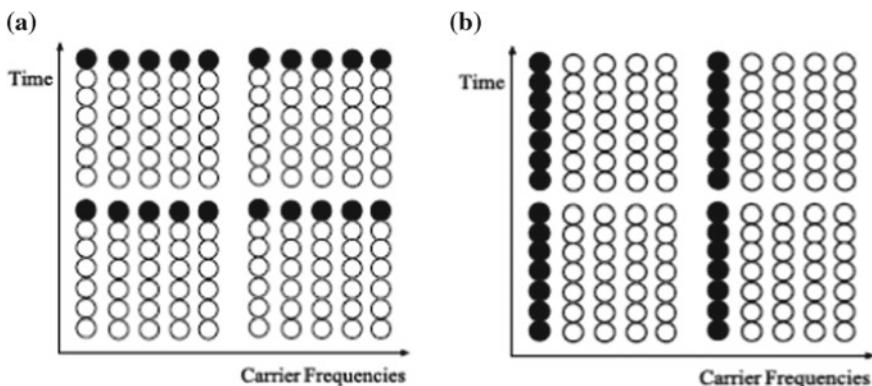


Fig. 11.3 **a** Block pilot, **b** comb pilot

$$S = \begin{bmatrix} s(0) & 0 & \cdots & 0 \\ 0 & s(1) & \cdots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & \cdots & 0 & s(N-1) \end{bmatrix} \quad (11.5)$$

where $s(k)$ denotes a pilot tone at the k th subcarrier, with $E\{s(k)\} = 0$, $\text{Var}\{s(k)\} = \sigma_s^2$, $k = 0, 1, 2, \dots, N - 1$. For a given channel gain $H(k)$ corresponding to the k th subcarrier, the received training signal $Y(k)$ can be represented as

$$\begin{aligned} Y &\triangleq \begin{bmatrix} Y(0) \\ Y(1) \\ \vdots \\ Y(N-1) \end{bmatrix} = \begin{bmatrix} s(0) & 0 & \cdots & 0 \\ 0 & s(1) & \vdots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ 0 & \cdots & 0 & s(N-1) \end{bmatrix} \begin{bmatrix} H(0) \\ H(1) \\ \vdots \\ H(N-1) \end{bmatrix} \\ &+ \begin{bmatrix} \eta(0) \\ \eta(1) \\ \vdots \\ \eta(N-1) \end{bmatrix} = SH + \eta \end{aligned} \quad (11.6)$$

where η is a noise vector with $E\{\eta(k)\} = 0$, $\text{Var}\{\eta(k)\} = \sigma_\eta^2$, and $k = 0, 1, 2, \dots, N - 1$.

11.3.1 LS Channel Estimation

The LS is a well-known method and widely used for estimation due to its simplicity. LS channel estimate is represented by:

$$\hat{H}_{\text{LS}} = S^{-1}Y \quad (11.7)$$

The pilot subcarriers are interpolated to estimate the channel for data symbols.

11.3.2 DFT-Based Channel Estimation

The DFT-based channel estimation technique improves the performance of LS channel estimation by removing the effect of noise outside the maximum channel delay [8].

The IDFT of the channel estimate $\{\widehat{H}(k)\}_{k=0}^{N^1}$ is written as

$$\text{IDFT}\{\widehat{H}(k)\} = h(n) + \eta(n) \triangleq \hat{h}(n), \quad n = 0, 1, \dots, N - 1 \quad (11.8)$$

where $\widehat{H}(k)$ is the estimate of the channel H at the k th subcarrier, obtained by LS, and $\eta(n)$ denotes the noise component. If the maximum channel delay is δC_d then

$$\hat{h}_{\text{DFT}}(n) = \begin{cases} h(n) + \eta(n), & n = 0, 1, \dots, \delta C_d - 1 \\ 0, & \text{otherwise} \end{cases} \quad (11.9)$$

and transformed back to the frequency domain as follows:

$$\widehat{H}_{\text{DFT}}(n) = \text{DFT}\{\hat{h}_{\text{DFT}}(n)\} \quad (11.10)$$

11.3.3 MIMO-OFDM Channel Estimation Using LS

Using Eq. (11.7), the LS estimate of the channel between j th transmitter and i th receiver antenna for MIMO-OFDM system can be expressed as

$$\widehat{H}_{\text{LS}}^{(j,i)} = (s^{(j)})^{-1} Y^{(i)} \quad (11.11)$$

$s^{(j)}$ is an $N \times N$ diagonal matrix with the pilots of the j th transmit antenna as diagonal elements and $Y^{(i)}$ is received vector of length N at receiver antenna i .

11.3.4 Channel Estimation Using MATLAB

The following MATLAB program is written using built-in MATLAB function “interpolation” to evaluate the MSE performance of LS and LS-DFT methods for channel estimation. For different E_b/N_0 s, the mean square errors (MSE) for LS and LS-DFT are shown in Fig. 11.4. From Fig. 11.4, it is observed that LS-DFT performs better than the LS method for channel estimation.

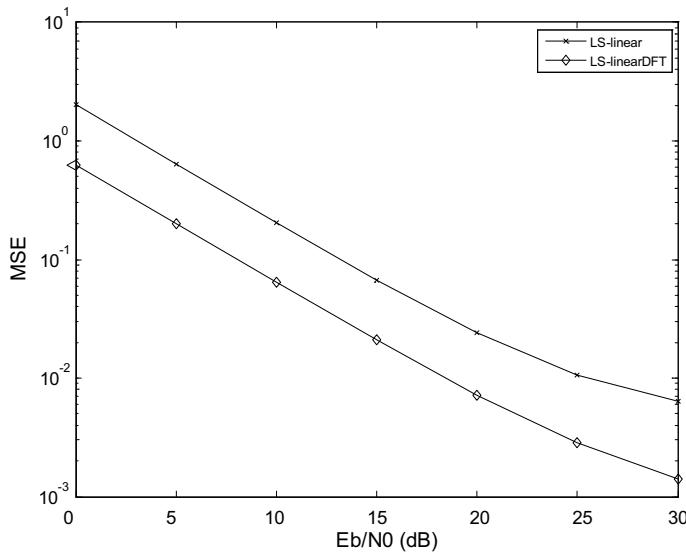


Fig. 11.4 E_b/N_0 versus MSE for channel estimation using LS and LS-DFT

Program 11.1 MATLAB program for Channel Estimation Using LS and LS-DFT methods

```
%channel_estimation.m
% for LS/DFT Channel Estimation with linear/spline interpolation
clear all; close all;
N_fft=32; N_g=N_fft/8; N_ofdm=N_fft+N_g; N_sym=100;
N_ps=4; N_p=N_fft/N_ps; N_d=N_fft-N_p; % Pilot spacing, Numbers of
pilots and data per OFDM symbol
N_bps=4; M=2^N_bps; % Number of bits per (modulated) symbol
mod_object = modem.qammod('M',M, 'SymbolOrder','gray');
demod_object = modem.qamdemod('M',M, 'SymbolOrder','gray');
Es=1; A=sqrt(3/2*(M-1)*Es); % Signal energy& QAM normalization factor
EbN0s = [0:5:30]; sq2=sqrt(2);
for i=1:length(EbN0s)
    EbN0 = EbN0s(i);
    rand('seed',1); randn('seed',1);
    MSE_LSi = 0; MSE_DFTi=0;
    for nsym=1:N_sym
        X_p = 2*(randn(1,N_p)>0)-1; % Pilot sequence generation
        msg_int=randint(1,N_fft-N_p,M); % bit generation
        Data = modulate(mod_object,msg_int)*A;
        ip = 0; pilot_loc = [];
        for k=1:N_fft
            if mod(k,N_ps)==1
                X(k) = X_p(floor(k/N_ps)+1); pilot_loc = [pilot_loc k]; ip = ip+1;
            else
                X(k) = Data(k-ip);
            end
        end
    end
end
```

```

x = ifft(X,N_fft); % IFFT
xt = [x(N_fft-N_g+1:N_fft) x]; % Add CP
h = [(randn+j*randn) (randn+j*randn)/2]; % generates a (2-tap) channel
H = fft(h,N_fft); channel_length = length(h); % True channel and its
time-domain lengt
y_channel = conv(xt, h); % Channel path (convolution)
yt = awgn(y_channel,EbN0,'measured');
y = yt(N_g+1:N_ofdm); % Remove CP
Y = fft(y); % FFT
k=1:N_p; Est_LS(k) = Y(pilot_loc(k))./X_p(k); % LS channel estimation
Est_HLS = interpolate(Est_LS,pilot_loc,N_fft,'linear');
h_estLS = ifft(Est_HLS); h_DFT = h_estLS(1:channel_length);
Est_HDFT = fft(h_DFT,N_fft); % DFT-based channel estimation
MSE_LSi = MSE_LSi+ (H-Est_HLS)*(H-Est_HLS)';
MSE_DFTi = MSE_DFTi+ (H-Est_HDFT)*(H-Est_HDFT)';
end
MSE_LS(i)=MSE_LSi; MSE_DFT(i)=MSE_DFTi;
end
MSE_LS = MSE_LS/(N_fft*N_sym);
MSE_DFT = MSE_DFT/(N_fft*N_sym);
figure(1), semilogy(EbN0s,MSE_LSi,'-x', EbN0s,MSE_DFT,'-d')
legend('LS-linear','LS-linearDFT')
xlabel ('Eb/N0 (dB)') ylabel ('MSE')

```

11.4 MIMO Channel Decomposition

A MIMO channel can be looked as a set of independent SISO channels using the singular value decomposition (SVD). The process requires precoding at the transmitter and receiver shaping at the receiver as shown in Fig. 11.5. This requires knowledge of the channel at the transmitter. The H matrix can be written in SVD form as

$$H = U \Sigma V^H \quad (11.12)$$

where U and V are unitary matrices ($U^H U = I_{N_R}$ and $V^H V = I_{N_T}$) and Σ is a $N_R \times N_T$ diagonal matrix of the singular values (σ_j) of H matrix. If H is a full rank matrix, there are $\min(N_R, N_T)$ of nonzero singular values and hence, with the same number of independent channels.

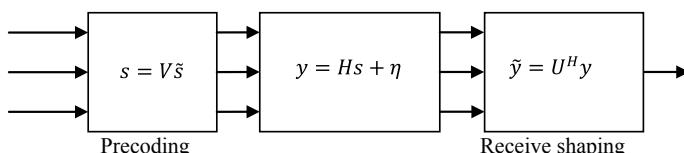


Fig. 11.5 Decomposition of a MIMO channel with full CSI

The received signal $\tilde{\mathbf{y}}$ is given by

$$\tilde{\mathbf{y}} = \mathbf{U}^H \mathbf{y} \quad (11.13)$$

The above equation can be rewritten as

$$\tilde{\mathbf{y}} = \mathbf{U}^H (\mathbf{H} \mathbf{s} + \boldsymbol{\eta}) \quad (11.14)$$

Now, substituting Eq. (11.12) in the above equation, we obtain

$$\tilde{\mathbf{y}} = \mathbf{U}^H (\mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^H \mathbf{s} + \boldsymbol{\eta}) \quad (11.15)$$

Since $\mathbf{s} = \mathbf{V} \tilde{\mathbf{s}}$, Eq. (11.15) can be rewritten as

$$\begin{aligned} \tilde{\mathbf{y}} &= \mathbf{U}^H (\mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^H \mathbf{V} \tilde{\mathbf{s}} + \boldsymbol{\eta}) \\ &= \mathbf{U}^H \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^H \mathbf{V} \tilde{\mathbf{s}} + \mathbf{U}^H \boldsymbol{\eta} \\ &= \boldsymbol{\Sigma} \tilde{\mathbf{s}} + \tilde{\boldsymbol{\eta}} \end{aligned} \quad (11.16)$$

From Eq. (11.16), it can be observed that the output is the product of precoded input signal $\tilde{\mathbf{s}}$ and the singular value matrix $\boldsymbol{\Sigma}$. The distribution of the noise does not change by multiplying the noise $\boldsymbol{\eta}$ by the unitary matrix \mathbf{U}^H .

Hence, Eq. (11.16) can be explicitly can be rewritten as

$$\begin{bmatrix} \tilde{y}_1 \\ \tilde{y}_2 \\ \vdots \\ \tilde{y}_{N_T} \end{bmatrix} = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{N_T} \end{bmatrix} \begin{bmatrix} \tilde{s}_1 \\ \tilde{s}_2 \\ \vdots \\ \tilde{s}_{N_T} \end{bmatrix} + \begin{bmatrix} \tilde{\eta}_1 \\ \tilde{\eta}_2 \\ \vdots \\ \tilde{\eta}_{N_T} \end{bmatrix} \quad (11.17)$$

where $\sigma_1, \sigma_1, \dots, \sigma_{N_T}$ are the eigenvalues.

The above system can be written in decoupled form as

$$\tilde{y}_1 = \sigma_1 \tilde{s}_1 + \tilde{\eta}_1 \quad (11.18a)$$

$$\tilde{y}_2 = \sigma_2 \tilde{s}_2 + \tilde{\eta}_2 \quad (11.18b)$$

⋮

$$\tilde{y}_{N_T} = \sigma_{N_T} \tilde{s}_{N_T} + \tilde{\eta}_{N_T} \quad (11.18c)$$

Thus, the above equivalent system represents the parallelization of MIMO channel with N_T information streams transmitted in parallel.

If we consider i th channel,

$$\tilde{y}_i = \sigma_i \tilde{s}_i + \tilde{\eta}_i \quad (11.19)$$

The SNR of the system is given as

$$\sigma_i^2 \frac{E[|s_i|^2]}{\sigma_n^2} = \sigma_i^2 \frac{P_i}{\sigma_n^2} \quad (11.20)$$

where P_i is the power of i th data stream s_i , σ_n^2 is the noise power, and σ_i^2 is the power gain.

The Shannon capacity of the i th channel can be expressed by

$$C_i = \log_2 \left(1 + \sigma_i^2 \frac{P_i}{\sigma_n^2} \right) \quad (11.21)$$

Since the MIMO channel can be viewed as N_T parallel information streams, the capacity of MIMO channel can be given by sum of the individual capacities as

$$C = \sum_{i=1}^{N_T} \log_2 \left(1 + \sigma_i^2 \frac{P_i}{\sigma_n^2} \right) \quad (11.22)$$

Example 11.1 Find a parallel channel model for a MIMO system, the H matrix of which is given by

$$H = \begin{bmatrix} 4 & -12 & 0 \\ 6 & 8 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

Solution

$$\begin{aligned} &= \begin{bmatrix} \frac{4}{\sqrt{52}} & -\frac{12}{\sqrt{208}} & 0 \\ \frac{6}{\sqrt{52}} & \frac{8}{\sqrt{208}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} -\frac{12}{\sqrt{208}} & \frac{4}{\sqrt{52}} & 0 \\ \frac{8}{\sqrt{208}} & \frac{6}{\sqrt{52}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{208} & 0 & 0 \\ 0 & \sqrt{52} & 0 \\ 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

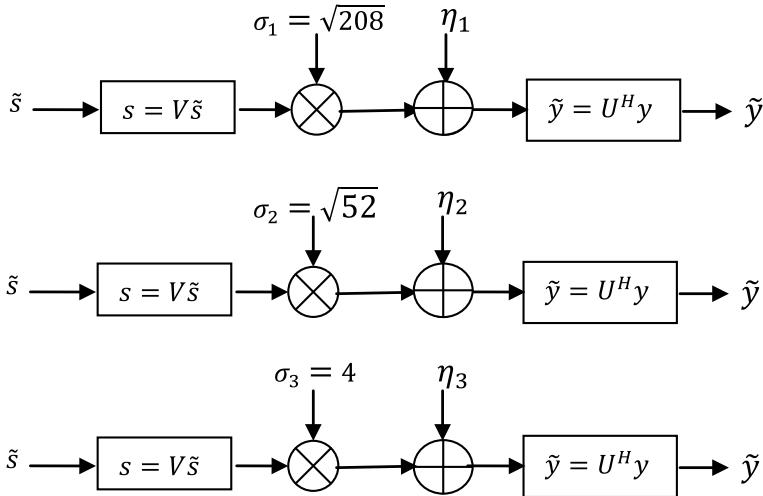


Fig. 11.6 SVD decomposition of a matrix channel into three independent SISO channels

The center matrix Σ contains the singular values (σ_j) of the H matrix. The rank of the matrix is equal to the number of singular values. This process decomposes the matrix channel into three independent SISO channels, with gains of $\sqrt{208}$, $\sqrt{52}$, and 4, respectively, as shown in Fig. 11.6. The number of significant eigenvalues specifies the maximum degree of diversity. The larger a particular eigenvalue, the more reliable is that channel. The most important benefit of the SVD approach is that it allows for enhanced array gain—the transmitter can send more power over the better channels and less (or no) power over the worst ones. Thus, the first channel with the gain of $\sqrt{208}$ will have better performance than the other two. The number of principle components is a measure of the maximum degree of diversity that can be realized in this way.

Example 11.2 Find a parallel channel model for a MIMO system, the H matrix of which is given by

$$H = \begin{bmatrix} 0.4 + j0.6 & j & 2 \\ -0.8 & 0.4 + j0.2 & 1.5 - j0.6 \\ j0.6 & -0.7 & -0.1 + j1.1 \end{bmatrix}$$

Solution The SVD decomposition using MATLAB gives

$$U = \begin{bmatrix} -0.4390 - j0.6062 & 0.2203 + j0.3263 & 0.2389 - j0.4773 \\ -0.4426 - j0.2417 & -0.8023 - j0.0633 & -0.2637 + j0.1685 \\ 0.3571 - j0.2408 & 0.0817 + j0.4366 & -0.7817 - j0.0777 \end{bmatrix}$$

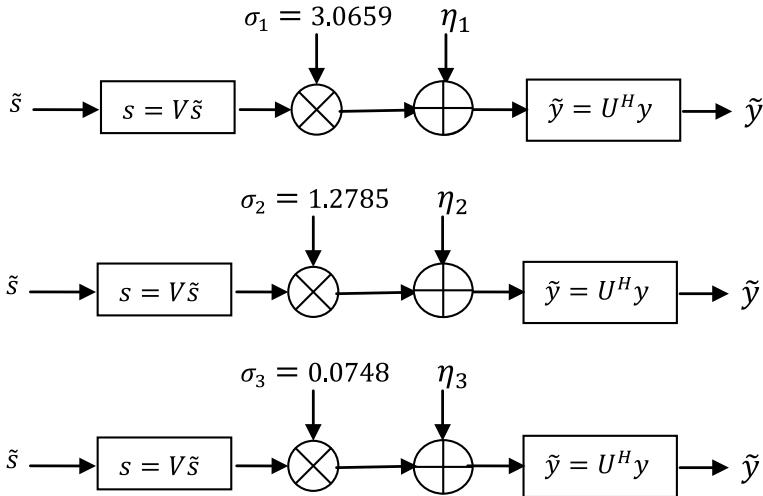


Fig. 11.7 SVD decomposition of a matrix channel into three independent SISO channels

$$\Sigma = \begin{bmatrix} 3.0659 & 0 & 0 \\ 0 & 1.2785 & 0 \\ 0 & 0 & 0.0748 \end{bmatrix}$$

$$V = \begin{bmatrix} -0.1075 & 0.9289 & -0.3543 \\ -0.3528 + j0.1955 & -0.0505 - j0.3056 & -0.0252 - j0.8607 \\ -0.5537 - j0.7206 & -0.1978 - j0.0449 & -0.3505 + j0.1011 \end{bmatrix}$$

See Fig. 11.7.

Optimal MIMO Capacity

The optimal power allocation problem can be formulated as

$$C_{\max} = \sum_{i=1}^{N_T} \log_2 \left(1 + \sigma_i^2 \frac{P_i}{\sigma_n^2} \right) \quad (11.23)$$

Subject to the condition $\sum_{i=1}^{N_T} P_i < P$.

Employing the Lagrange-multiplier-based technique to the above constrained optimization problem, the Lagrangian cost function can be formulated as

$$f \left([P_1, P_2, \dots, P_{N_T}]^T, \mu \right) = \sum_{i=1}^{N_T} \log_2 \left(1 + \sigma_i^2 \frac{P_i}{\sigma_n^2} \right) + \mu \left(P - \sum_{i=1}^{N_T} P_i \right) \quad (11.24)$$

where μ is the Langrange multiplier.

Taking derivative of $f\left(\left[P_1, P_2, \dots, P_{N_T}\right]^T, \mu\right)$ with respect to P_i and equating to zero, we obtain

$$\frac{\partial}{\partial P_i} f\left(\left[P_1, P_2, \dots, P_{N_T}\right]^T, \mu\right) = 0 \quad (11.25)$$

$$\frac{\frac{\sigma_i^2}{\sigma_n^2}}{1 + \sigma_i^2 \frac{P_i}{\sigma_n^2}} - \mu = 0 \quad (11.26)$$

The solution of the above equation yields

$$\frac{\sigma_i^2}{\sigma_n^2} \frac{1}{\mu} = 1 + \sigma_i^2 \frac{P_i}{\sigma_n^2} \quad (11.27)$$

where $P_i = \left(\frac{1}{\mu} - \frac{\sigma_n^2}{\sigma_i^2}\right)^+$

The function $(x)^+$ is defined by

$$\begin{aligned} (x)^+ &= x \text{ for } x \geq 0 \\ &= 0 \text{ for } x < 0 \end{aligned} \quad (11.28)$$

This is (x) due to the fact that the power can not be negative. The Lagrange multiplier μ can be determined from the constraint

$$\sum_{i=1}^{N_T} \left(\frac{1}{\mu} - \frac{\sigma_n^2}{\sigma_i^2} \right)^+ = P \quad (11.29)$$

The above optimal power allocation is the well-known water-filling power allocation algorithm which is illustrated in Fig. 11.8.

Figure 11.8 shows that more power must be allocated to the channel with higher SNR.

Furthermore, if an SNR is below the threshold given in terms of μ , no power is allocated to the corresponding channels. As can be seen from Fig. 10.7, power is not allocated to the channels 3 and 6.

Example 11.3 Assuming total power is 1 W, $\sigma_n^2 = 3$ dB, find the channel capacity and the optimal power allocation for MIMO channel of Example 10.1.

Example: $\sigma_n^2 = 10^{3/10} = 2$; $P = 1$; $\sigma_1^2 = 208$; $\sigma_2^2 = 52$; $\sigma_3^2 = 16$.

$$\begin{aligned} \left(\frac{1}{\mu} - \frac{\sigma_n^2}{\sigma_1^2} \right) + \left(\frac{1}{\mu} - \frac{\sigma_n^2}{\sigma_2^2} \right) + \left(\frac{1}{\mu} - \frac{\sigma_n^2}{\sigma_3^2} \right) &= \left(\frac{1}{\mu} - \frac{2}{208} \right) + \left(\frac{1}{\mu} - \frac{2}{52} \right) \\ &+ \left(\frac{1}{\mu} - \frac{2}{16} \right) = \left(\frac{1}{\mu} - \frac{1}{104} \right) + \left(\frac{1}{\mu} - \frac{1}{26} \right) + \left(\frac{1}{\mu} - \frac{1}{8} \right) = 1 \end{aligned}$$

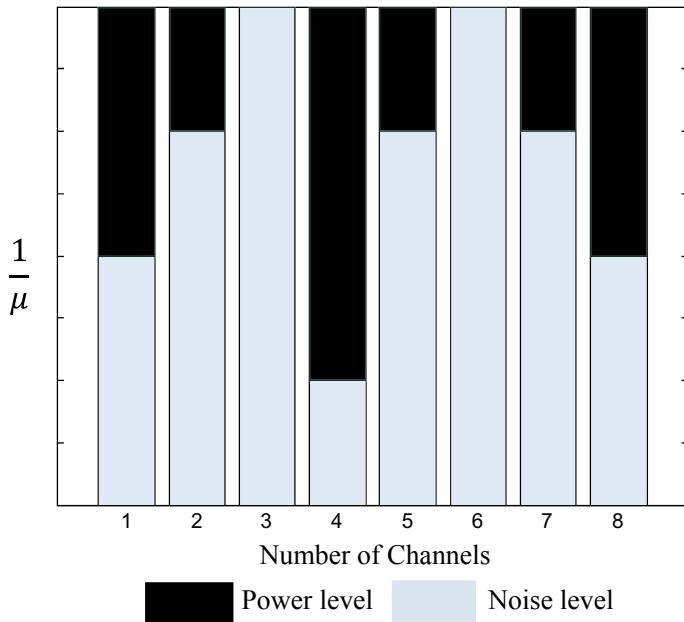


Fig. 11.8 Water-filling power allocation algorithm

$$\frac{1}{\mu} = \frac{1 + \frac{1}{104} + \frac{1}{26} + \frac{1}{8}}{3} = 0.391$$

$$P_1 = 0.391 - \frac{1}{104} = 0.3814 \text{ W}$$

$$P_2 = 0.391 - \frac{1}{26} = 0.3525 \text{ W}$$

$$P_3 = 0.391 - \frac{1}{8} = 0.266 \text{ W}$$

$$\begin{aligned}
 C &= \log_2 \left(1 + \sigma_1^2 \frac{P_1}{\sigma_n^2} \right) + \log_2 \left(1 + \sigma_2^2 \frac{P_2}{\sigma_n^2} \right) + \log_2 \left(1 + \sigma_3^2 \frac{P_3}{\sigma_n^2} \right) \\
 &= \log_2 \left(1 + 208 \frac{0.3814}{2} \right) + \log_2 \left(1 + 52 \frac{0.3525}{2} \right) + \log_2 \left(1 + 16 \frac{0.266}{2} \right) \\
 &= \log_2 (1 + 104 \times 0.3814) + \log_2 (1 + 26 \times 0.3525) + \log_2 (1 + 8 \times 0.266) \\
 &= 10.3365 \text{ b/s/Hz}
 \end{aligned}$$

11.5 MIMO Channel Capacity

Let s and y be N_T and N_R length vectors containing the transmitted and received symbols, respectively, for a MIMO system with N_T transmit and N_R receive antennas. Then, the received signal y can be rewritten in a matrix form as follows:

$$y = \sqrt{\frac{E_s}{N_T}} H s + \eta \quad (11.30)$$

where

$$\begin{aligned} y &= [y_1 y_2 \dots y_{N_R}] \\ s &= [s_1 s_2 \dots s_{N_T}] \\ \eta &= [\eta_1 \eta_2 \dots \eta_{N_R}] \end{aligned}$$

E_s is the total energy of N_T symbols transmitted

11.5.1 Capacity of Deterministic MIMO Channel When CSI Is Known to the Transmitter

The capacity of a deterministic channel is defined by Shannon as

$$C = \max_{f(s)} I(s, y) \text{ bits/channel use} \quad (11.31)$$

$I(s, y)$ is called the mutual information of s and y . The capacity of the channel is the maximum information that can be transmitted from s to y by varying the channel probability density function (*pdf*). $f(s)$ is the *pdf* of the transmit signal s . From information theory, we get the relationship of mutual information between two random variables as a function of their entropy as

$$I(s, y) = H(y) - H(y|s) \quad (11.32)$$

$$H(y|s) = H(\eta) \quad (11.33)$$

Using Eqs. (11.32) and (11.31) can be rewritten as

$$I(s, y) = H(y) - H(\eta) \quad (11.34)$$

The second term is constant for a deterministic channel because it is a function of noise. Hence, mutual information is maximum only when the term $H(y)$ is maximum.

Using Eq. (11.30), the auto-correlation matrix of y can be written as

$$\begin{aligned}
 R_{yy} &= E[yy^H] = E\left[\left(\sqrt{\frac{E_s}{N_T}} Hs + \eta\right)\left(\sqrt{\frac{E_s}{N_T}} Hs + \eta\right)^H\right] \\
 &= E\left[\left(\frac{E_s}{N_T} Hss^H H^H + \eta\eta^H\right)\right] \\
 &= \frac{E_s}{N_T} E[Hss^H H^H + \eta\eta^H] \\
 &= \frac{E_s}{N_T} HE[ss^H]H^H + E[\eta\eta^H] \\
 &= \frac{E_s}{N_T} HR_{ss}H^H + N_o I_{N_R}
 \end{aligned} \tag{11.35}$$

where R_{ss} is the auto-correlation of the transmitted signal vector s and N_o is the power spectral density of the additive noise $\{\eta_i\}_{i=1}^{N_R}$. The entropy $H(y)$ is maximized when both s and y are zero mean circular symmetric complex Gaussian (ZMCSCG) random variables. Then, the $H(y)$ and $H(\eta)$ are given by

$$H(y) = \log_2 \{\det(\pi e R_{yy})\} \tag{11.36}$$

$$H(\eta) = \log_2 \{\det(\pi e N_o I_{N_R})\} \tag{11.37}$$

Using Eqs. (11.36) and (11.37), it is shown in [9] that the mutual information given by Eq. (11.34) can be expressed as

$$I(s, y) = \log_2 \det \left\{ I_{N_R} + \frac{E_s}{N_T N_o} H R_{ss} H^H \right\} \text{bits/s/Hz} \tag{11.38}$$

Since $SNR = \frac{E_s}{N_o}$, Eq. (11.38) can be rewritten as

$$I(s, y) = \log_2 \det \left\{ I_{N_R} + \frac{SNR}{N_T} H R_{ss} H^H \right\} \text{bits/s/Hz} \tag{11.39}$$

From the above equation, we can write the expression for capacity as

$$C = I(s, y) = \max_{Tr(R_{ss})=N_T} \log_2 \det \left\{ I_{N_R} + \frac{SNR}{N_T} H R_{ss} H^H \right\} \tag{11.40}$$

It should be noted here that trace of R_{ss} matrix, $Tr(R_{ss}) = N_T$, when the transmission power for each transmit antenna is assumed to be 1.

11.5.2 Deterministic MIMO Channel Capacity When CSI Is Unknown at the Transmitter

When H is not known at the transmitter side, we can assume equal power distribution among the transmitters, R_{ss} is an identity matrix, that is, $R_{ss} = I_{N_T}$ and Eq. (11.40) becomes

$$C = \log_2 \det \left\{ I_{N_T} + \frac{\text{SNR}}{N_R} H H^H \right\} \quad (11.41)$$

This is the capacity equation for the MIMO channels with equal power. It should be noted that for a large number of transmit antennas and a fixed number of receive antennas, the law of large numbers yields

$$\lim_{N_T \rightarrow \infty} \frac{1}{N_R} H H^H = I_{N_T} \quad (11.42)$$

Thus, the MIMO channel capacity for large N_T becomes

$$C = N_R \log_2 \det \{ I_{N_T} + \text{SNR} \} \quad (11.43)$$

Example 11.4 Given the following $(3 \times 3$ MIMO) channel, find the capacity of this channel, when CSI is known at the receiver and unknown at the transmitter, SNR = 10 dB and bandwidth equal to 1 kHz. Compare this capacity calculation to that using SVD.

$$H = \begin{bmatrix} 0.4 + j0.6 & j & 2 \\ -0.8 & 0.4 + j0.2 & 1.5 - j0.6 \\ j0.6 & -0.7 & -0.1 + j1.1 \end{bmatrix}$$

Solution

$$\begin{aligned} H H^H &= \begin{bmatrix} 5.52 & 2.88 + j1.12 & 0.16 - j3.14 \\ 2.88 - j1.12 & 3.45 & -1.09 - j1.25 \\ 0.16 + j3.14 & -1.09 + j1.25 & 2.07 \end{bmatrix} \\ C &= B \log_2 \left(\det \left\{ I_{N_R} + \frac{\text{SNR}}{N_T} H H^H \right\} \right) \\ &= 7.7306 \text{ kbps} \end{aligned}$$

The singular values are equal to: 3.0659, 1.2785, and 0.0748.

The sum of the capacity of the three independent channels is equal to the same quantity as above equation.

$$C = B(\log_2(1 + 3.0659^2 \cdot 3.33) + \log_2(1 + 1.2785^2 \cdot 3.33) + \log_2(1 + 0.0748^2 \cdot 3.33)) = 7.7306 \text{ kbps}$$

Example 11.5 Assuming total power is 1 W, noise power is equal to 0.1 W, and the signal bandwidth is 50 kHz, find the channel capacity and optimal power allocation for MIMO channel, the H matrix of which is given by

$$H = \begin{bmatrix} 0.8 & 0.5 - j0.2 & 0.3 + j0.6 \\ 0.4 - j0.6 & 1.0 - j0.1 & 0.2 - j0.9 \\ 0.5 + j0.3 & 0.5 + j1.5 & 0.6 + j1.2 \end{bmatrix}$$

Solution The SVD decomposition using MATLAB gives

$$U = \begin{bmatrix} -0.3031 + j0.0117 & 0.8311 - j0.0717 & 0.4556 + j0.0668 \\ -0.2299 + j0.4705 & -0.2603 + j0.4833 & 0.4543 - j0.4670 \\ -0.6244 - j0.4938 & 0.0158 + j0.0499 & -0.3519 - j0.4896 \end{bmatrix}$$

$$S = \begin{bmatrix} 2.6687 & 0 & 0 \\ 0 & 0.9684 & 0 \\ 0 & 0 & 0.5834 \end{bmatrix}$$

$$V = \begin{bmatrix} -0.4036 & 0.3033 & 0.8632 \\ -0.5560 + j0.4056 & 0.2106 + j0.6082 & -0.3340 - j0.0240 \\ -0.5698 + j0.1969 & -0.2183 - j0.6680 & -0.1897 + j0.3267 \end{bmatrix}$$

The singular values computed for the three channels are $\sigma_1 = 2.6687$, $\sigma_2 = 0.9684$, and $\sigma_3 = 0.5834$. The SNR values for each channel assuming equal power allocation are

$$\begin{aligned} \sigma_n^2 &= 0.1; P = 1; \sigma_1^2 = 7.122; \sigma_2^2 = 0.9378; \sigma_3^2 = 0.3404. \\ \left(\frac{1}{\mu} - \frac{\sigma_n^2}{\sigma_1^2}\right) + \left(\frac{1}{\mu} - \frac{\sigma_n^2}{\sigma_2^2}\right) + \left(\frac{1}{\mu} - \frac{\sigma_n^2}{\sigma_3^2}\right) \\ &= \left(\frac{1}{\mu} - \frac{0.1}{7.122}\right) + \left(\frac{1}{\mu} - \frac{0.1}{0.9378}\right) + \left(\frac{1}{\mu} - \frac{0.1}{0.3404}\right) \\ &= \left(\frac{1}{\mu} - \frac{1}{71.22}\right) + \left(\frac{1}{\mu} - \frac{1}{9.378}\right) + \left(\frac{1}{\mu} - \frac{1}{3.404}\right) = 1 \\ \frac{1}{\mu} &= \frac{1 + \frac{1}{71.22} + \frac{1}{9.378} + \frac{1}{3.404}}{3} = 0.4715 \end{aligned}$$

Since the third channel with its SNR of 0.3404 is less than this threshold value of SNR, we do not allocate any power to the third channel and redo the calculations based only on the first two channels

$$\frac{1}{\mu} = \frac{1 + \frac{1}{71.22} + \frac{1}{9.378}}{2} = 0.5603$$

$$P_1 = 0.5603 - \frac{1}{71.22} = 0.5463 \text{ W}$$

$$P_2 = 0.5603 - \frac{1}{9.378} = 0.4537 \text{ W}$$

$$P_3 = 0 \text{ W}$$

$$\begin{aligned} C &= \log_2 \left(1 + \sigma_1^2 \frac{P_1}{\sigma_n^2} \right) + \log_2 \left(1 + \sigma_2^2 \frac{P_2}{\sigma_n^2} \right) \\ &= 50,000 \times \left[\log_2 \left(1 + 7.122 \frac{0.5463}{0.1} \right) + \log_2 \left(1 + 0.9378 \frac{0.4537}{0.1} \right) \right] \\ &= 50,000 \times \log_2 (1 + 71.22 \times 0.5463) + 50,000 \times \log_2 (1 + 9.378 \times 0.4537) \\ &= 368.555 \text{ kb/s} \end{aligned}$$

11.5.3 Random MIMO Channel Capacity

11.5.3.1 Random MIMO Channel Capacity with CSI Known at the Transmitter

In general, the MIMO channels are varying randomly. Hence, H is a random matrix and its channel capacity is also randomly time-varying. In practice, assuming that the random channel is an ergodic process, the MIMO channel capacity can be expressed by

$$C_{\text{erg}} = E \left[\max_{Tr(R_{ss})=N_T} \log_2 \det \left\{ I_{N_R} + \frac{\text{SNR}}{N_T} H R_{ss} H^H \right\} \right] \quad (11.44)$$

where the subscript erg stands for ergodic.

11.5.3.2 Random MIMO Channel Capacity with CSI Unknown at the Transmitter

When CSI is unknown at the transmitter, from Eq. (11.44), the ergodic capacity of the random MIMO channel is given by

$$C_{\text{erg}} = E \left[\log_2 \det \left\{ I_{N_R} + \frac{\text{SNR}}{N_T} \mathbf{H} \mathbf{H}^H \right\} \right] \quad (11.45)$$

Equation (11.45) can be written in terms of the positive eigenvalues as [8]

$$C_{\text{erg}} = E \left[\log_2 \det \left\{ I_{N_R} + \frac{\text{SNR}}{N_T} \lambda_i \right\} \right] \quad (11.46)$$

which is frequently known as an ergodic channel capacity.

Program 11.2 MATLAB program for ergodic capacity

```
clear all, close all
SNRdB = [0:35];
iter=1000;
C1=ergcap(SNRdB,1,1,iter);
C2=ergcap(SNRdB,2,2,iter);
C3=ergcap(SNRdB,4,4,iter);
figure, plot(SNRdB,C1(:,1),'b-o', SNRdB,C2(:,1),'b-*', SNRdB,C3(:,1),'b-+');
xlabel('SNR(dB)'); ylabel('bps/Hz'); set(gca,'fontsize',10);
legend('{\it N_T}=1,{\it N_R}=1','{\it N_T}=2,{\it N_R}=2','{\it N_T}=4,{\it N_R}=4')
```

The ergodic capacity of i.i.d random MIMO channel with CSI unknown at the transmitter with different transmitting and receiving antennas is shown in Fig. 10.8.

Program 11.3 MATLAB Function program for ergcap

```
function C=ergcap(SNRdB,nT,nR,iter)
n=min(nT,nR); I = eye(n);SNRlin=10.^((SNRdB/10.));
C(1,:)= zeros(1,length(SNRdB));
for ii=1:iter
    H = sqrt(0.5)*(randn(nR,nT))+j*randn(nR,nT);
    if nR>=nT, HH = H.*H; else HH = H.*H'; end
    for j=1:length(SNRdB) %random channel generation
        C(1,j)= C(1,j)+log2(real(det(I+SNRlin(j)/nT*HH)));
    end
end
C = C/iter;
```

From Fig. 11.9, it is observed that the number of the transmitting and the receiving antennas increases, the ergodic capacity increases.

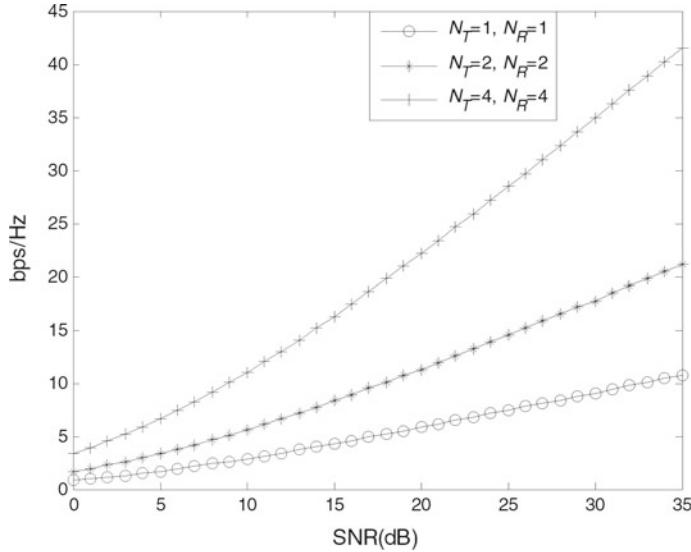


Fig. 11.9 Ergodic capacity of i.i.d random MIMO channel with CSI unknown at the transmitter

11.5.3.3 Capacity of Correlated Random MIMO Channel

In general, the elements of H are correlated by an amount that depends on the propagation environment as well as the polarization of the antenna elements and spacing between them [9]. One possible model for H that takes the fading correlation into account, splits the fading correlation into two independent components called receive correlation and transmit correlation, respectively [10, 11]. This amounts to modeling H as follows

$$H = R_r^{1/2} H_w R_t^{1/2} \quad (11.47)$$

where R_t and R_r are the transmit correlation and the receive correlation matrices, respectively, H_w is a matrix with independent Gaussian elements with unity variance, and the superscript 1/2 stands for the Hermitian square root of a matrix. The matrix R_r determines the correlation between the rows of H , and the matrix R_t determines the correlation between the columns of H . The diagonal entries of R_t and R_r are constrained to be a unity. The correlation matrices R_t and R_r can be measured or computed by assuming the scattering distribution around the transmit and receive antennas. For uniform linear array at the transmitter and the receiver, the correlation matrices R_t and R_r can be calculated according to two different methods given in [12, 13]. From [13], we have the following Toeplitz structure correlation matrices:

$$R_t = \begin{bmatrix} 1 & r_t & \cdots & r_t^{(N_T-1)^2} \\ r_t & 1 & \cdots & \cdots \\ \vdots & \cdots & \cdots & \cdots \\ r_t^{(N_T-1)^2} & \cdots & \cdots & 1 \end{bmatrix};$$

$$R_r = \begin{bmatrix} 1 & r_r & \cdots & r_r^{(N_R-1)^2} \\ r_r & 1 & \cdots & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ r_r^{(N_R-1)^2} & \cdots & \cdots & 1 \end{bmatrix} \quad (11.48)$$

where R_t and R_r represent $r(d_t)$ and $r(d_r)$, respectively, and $r(d)$ is the approximation for the fading correlation between two adjacent antenna elements averaged over all possible orientations of the two antennas in a given wave-field which can be expressed as [14]

$$r(d) \approx \exp(-23 \wedge^2 d^2) \quad (11.49)$$

where, d is the distance in wavelengths between two antennas and \wedge is the angular spread.

From Eq. (11.45), then, the MIMO channel capacity is given as

$$C = \log_2 \det \left\{ I_{N_R} + \frac{\text{SNR}}{N_T} R_r^{1/2} H_w R_t H_w^H R_r^{H/2} \right\} \quad (11.50)$$

The following examples demonstrate the performance of the correlated random MIMO channels with and without CSI known at the transmitter using MATLAB.

Example 11.6 Compare the capacity of spatially correlated random 4×4 channels with unknown CSI at the transmitter for a non-uniform antenna array structure with the following correlation matrices [13]

$$R_t = \begin{bmatrix} 1 & 0.3169 & 0.3863 & 0.0838 \\ 0.3169 & 1 & 0.7128 & 0.5626 \\ 0.3863 & 0.7128 & 1 & 0.5354 \\ 0.0838 & 0.5626 & 0.5354 & 1 \end{bmatrix};$$

$$R_r = \begin{bmatrix} 1 & 0.1317 & 0.1992 & 0.2315 \\ 0.1317 & 1 & 0.1493 & 0.1907 \\ 0.1992 & 0.1493 & 1 & 0.1996 \\ 0.2315 & 0.1907 & 0.1996 & 1 \end{bmatrix}$$

and an uniform antenna array structure with $r_t = r_r = 0.2$.

Program 11.4 MATLAB program for random MIMO channel capacity

```
%Program to compute the capacity of random MIMO channels with out CSI
with uniform and non-uniform correlated matrices
clear all, close all
SNRdB = [0:25];iter=1000; nT=4; nR=4;
n=min(nT,nR); I = eye(n);SNRlin=10.^{SNRdB/10.};rho=0.2;
Rtx=[1 0.3169 0.3863 0.0838; 0.3169 1 0.7128 0.5626;
      0.3863 0.7128 1 0.5354; 0.0838 0.5626 0.5354 1];
Rrx=[1 0.1317 0.1992 0.2315; 0.1317 1 0.1493 0.1907;
      0.1992 0.1493 1 0.1996; 0.2315 0.1907 0.1996 1];
Rtxu=[1 rho rho^2 rho^3; rho 1 rho rho^2;
      rho^2 rho 1 rho; rho^3 rho^2 rho 1];
Rrxu= Rtxu;
C1(1,:)= zeros(1,length(SNRdB)); C2(1,:)= zeros(1,length(SNRdB));
for ii=1:iter
    Hw = sqrt(0.5)*(randn(nR,nT)+j*randn(nR,nT));
    H = Rrx.^{1/2}*Hw*Rtx.^{1/2}; tmp = H.*H/nT;
    Hu = Rrxu.^{1/2}*Hw*Rtxu.^{1/2}; tmp1 = Hu.*Hu/nT;;
    for i=1:length(SNRdB) %random channel generation
        C2(1,i) = C2(1,i)+log2(real(det(I+SNRlin(i)*tmp1)));
        C1(1,i) = C1(1,i)+log2(real(det(I+SNRlin(i)*tmp)));
    end
end
C1 = C1/iter; C2=C2/iter;
figure, plot(SNRdB,C1(1,:),'-', SNRdB,C2(1,:),'--');
xlabel('SNR(dB)'); ylabel('bps/Hz'); set(gca,'fontsize',10);
legend('with non-uniform correlation matrices' , 'with uniform correlation
matrices')
```

The capacity of the random MIMO channels without CSI with uniform and non-uniform correlated matrices is shown in Fig. 11.10.

From Fig. 11.10, it is observed that the capacity of a random MIMO channel without CSI and with uniform correlated matrices gives the better performance than the non-uniform correlated matrices.

Example 11.7 Compare the capacity of spatially correlated random 4×4 channels with known and unknown CSI at the transmitter for an uniform antenna array structure with $r_t = r_r = 0.2$.

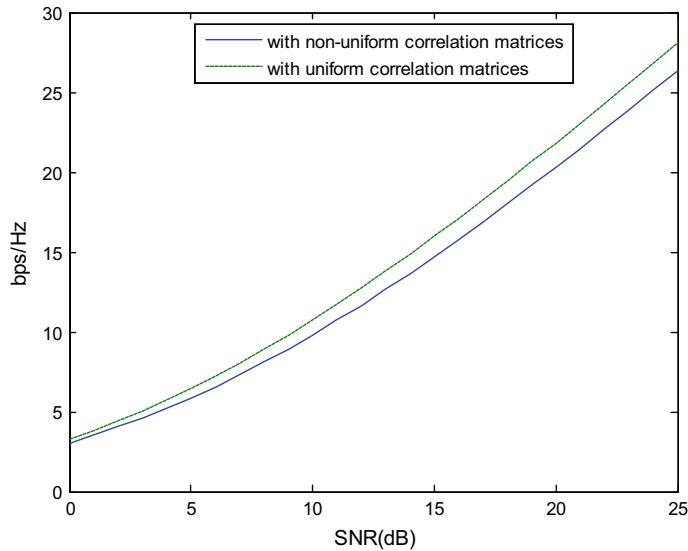


Fig. 11.10 Random MIMO channel capacity without CSI with uniform and non-uniform correlated matrices

Program 11.5 MATLAB program for spatially correlated 4×4 channel capacity

```
%Program to compute the capacity of random MIMO channels with and
without CSI with uniform correlated matrices
clear all; close all
SNRdB = [0:25]; iter=1000; nT=4; nR=4;
n=min(nT,nR); I = eye(n); SNRlin=10.^((SNRdB/10.);rho=0.2;
Rtx=[1 rho rho^2 rho^3; rho 1 rho rho^2;
rho^2 rho 1 rho; rho^3 rho^2 rho 1];
Rrx=Rtx;
C1(1,:) = zeros(1,length(SNRdB));
C2(1,:) = zeros(1,length(SNRdB));
for ii=1:iter
    Hw = sqrt(0.5)*(randn(nR,nT)+j*randn(nR,nT));
    H = Rrx^(1/2)*Hw*Rtx^(1/2);tmp = H'*H/nT;HH= H'*H; Lamda = svd(H'*H);
    for i=1:length(SNRdB) %random channel generation
        C1(1,i) = C1(1,i)+ log2(real(det(I+SNRlin(i)*tmp)));
        gama = WaterFilling(Lamda,SNRlin(i),nT);
        C2(1,i) = C2(1,i)+log2(real(det(I+(SNRlin(i)/nT)*diag( gama
)*diag(Lamda))));
    end
end
C1 = C1/iter; C2=C2/iter;
figure, plot(SNRdB,C1(1,:),'-', SNRdB,C2(1,:),'--');
xlabel('SNR(dB)'); ylabel('bps/Hz'); set(gca,'fontsize',10);
legend('with unknown CSI at the transmitter', 'with known CSI at the transmis-
ter')
```

Program 11.6 MATLAB Function program for Water Filling

```

function [gamat]=WaterFilling(Lamda,SNR,nT)
r=length(Lamda); gama = zeros(1,r);
index=[1:r]; indext=index;
p=1;
while p<r
    ir=[1:r-p+1];
    temp= sum(1./Lamda(indext(ir)));
    mu = nT/(r-p+1)*(1+1/SNR*temp);
    gama(indext(ir))=mu-nT./(SNR*Lamda(indext(ir)));
    if min(gama(indext))<0
        i=find(gama==min(gama)); ii=find(indext==i);
        indext1=[indext([1:ii-1]) indext([ii+1:end])];
        indext=indext1;
        p=p+1;
        clear gama;
    else
        p=r;
    end
end
gamat=zeros(1,length(Lamda)); gamat(indext)=gama(indext);

```

The Comparison of the capacity of spatially correlated channels with known and unknown CSI at the transmitter is shown in Fig. 11.11.

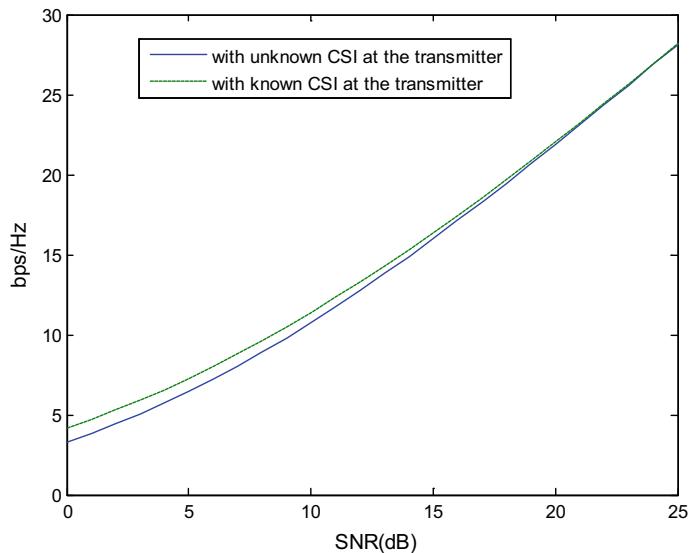


Fig. 11.11 Comparison of the capacity of spatially correlated 4×4 channels with known and unknown CSI at the transmitter

From Fig. 10.10, it is observed that, the capacity of spatially correlated 4×4 channels with known CSI at the transmitter gives better performance than with unknown CSI at the transmitter.

When $N_T = N_R$ and SNR is high, Eq. (11.50) can be approximated as

$$C = \log_2 \det \left\{ I_{N_R} + \frac{\text{SNR}}{N_T} H_w H_w^H \right\} + \log_2 \det(R_t) + \log_2 \det(R_r) \quad (11.51)$$

From Eq. (11.51), it can be observed that the MIMO channel capacity is reduced due to the correlation between the transmit and receive antennas and the reduction is

$$\log_2 \det(R_t) + \log_2 \det(R_r) \quad (11.52)$$

it is shown in [8] that the value in Eq. (11.52) is always negative by the fact that $\log_2 \det(R_t) \leq 0$ for any correlation matrix R .

The following example illustrates how correlation reduces the channel capacity using MATLAB.

Example 11.8 Compare the capacity of i.i.d and correlated random 2×2 channels with CSI unknown at the transmitter assuming

$R_t = \begin{bmatrix} 1 & 0.76\exp(0.17j\pi) \\ 0.76\exp(0.17j\pi) & 1 \end{bmatrix}$; R_r ; is a 2×2 identity matrix, i.e. no correlation exists between the receive antennas.

Program 11.7 MATLAB program for i.i.d and correlated MIMO channels capacity

```
%Program to compute the capacity of iid and correlated MIMO channels
clear all, close all;
SNRdB=[0:5:25]; SNRlin=10.^ (SNRdB/10);
iterations=1000; nT=2; nR=2;
n=min(nT,nR); I = eye(n); sq2=sqrt(0.5);
R=[1 0.76*exp(0.17j*pi) ; 0.76*exp(-0.17j*pi) 1 ];
Ciid=zeros(1,length(SNRdB)); Ccorr=zeros(1,length(SNRdB));
for iter=1:iterations
    H = sq2*(randn(nR,nT)+j*randn(nR,nT));
    Hc = H*R^(1/2);
    temp1 = H'*H/nT; temp2 = Hc'*Hc /nT;
    for i=1:length(SNRdB)
        Ciid(i) = Ciid(i) + log2(det(I+SNRlin(i)*temp1));
        Ccorr(i) = Ccorr(i) + log2(det(I+SNRlin(i)*temp2));
    end
    Ciid = real(Ciid)/iterations; Ccorr = real(Ccorr)/iterations;
    plot(SNRdB,Ciid, SNRdB, Ccorr,'.');
    xlabel('SNR (dB)'); ylabel('Capacity(bps/Hz)'); set(gca,'fontsize',10)
    legend('iid 2x2 fading channels','correlated 2x2 fading channels');
end
```

A Comparison of the capacity of i.i.d and correlated 2×2 channels is shown in Fig. 11.12. From Fig. 11.12, it is observed that, the capacity of the i.i.d with

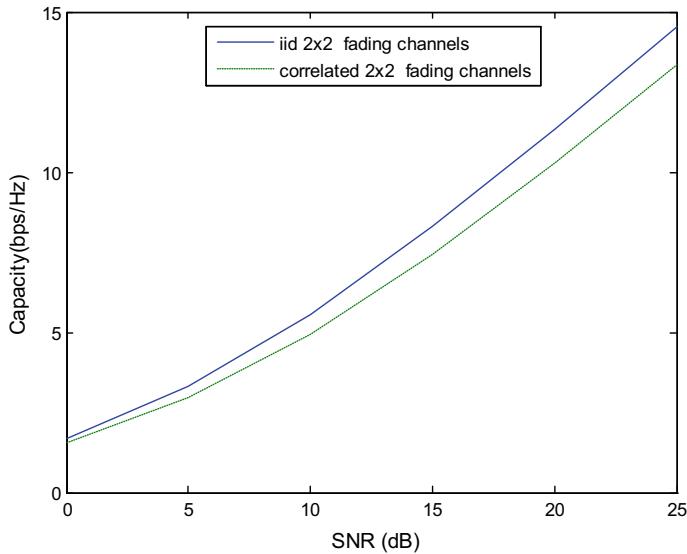


Fig. 11.12 Comparison of the capacity of i.i.d and correlated 2×2 channels

unknown CSI at the transmitter gives the better performance than correlated random 2×2 channels with unknown CSI at the transmitter.

11.6 MIMO-OFDM Channel Estimation Using OMP Algorithm

Coherent detection and equalization in multiple-input multiple-output orthogonal frequency division multiplexing (MIMO-OFDM) systems require channel state information (CSI) at the receiver as discussed in chapter. In real wireless environments, however, the CSI is not known. Therefore, channel estimation is of crucial importance in MIMO-OFDM systems.

For simplicity, the pilot-assisted MIMO-OFDM channel estimation is generally decomposed into simultaneous estimation of several SISO-OFDM channels. To obtain better estimation performance of every channel, all other antennas must remain silent, when the i th antenna transmits a pilot symbol namely, [15]

$$x_j(n) = 0(n \in \lambda_i, 1 \leq i, j \leq N_T), \text{ and } i \neq j \quad (11.53)$$

where $\lambda_i = \{k_1, k_2, \dots, k_{N_p}\}$ i.e. set of pilot location indices.

We consider a MIMO system with two transmit and two receive antennas where we estimate the MIMO-OFDM channel by OMP algorithm with QR factorization.

11.6.1 OMP Algorithm

Assume \hat{h} be the estimate of the channel coefficients. \wedge_k be the set of indices of the nonzero channel coefficients, r be the residual, and $\varphi_1, \varphi_2, \dots, \varphi_M$ be the columns of measurement matrix θ and i be the iteration count. The step-wise procedure is as below.

Algorithm 1 OMP Algorithm

Step 1: Initialize $r_0 = y$, $\wedge_0 = 0$ and $i = 1$

Step 2: Determine the new index λ_i by selecting the maximum absolute value of the correlation between θ^H and residual r_{i-1}

$$\lambda_i = \operatorname{argmax}_{j=1:N} |\theta^H r_{i-1}| \quad (11.54)$$

Step 3: Unite the newly selected index λ_i with the index \wedge_{i-1}

$$\wedge_i = \wedge_{i-1} \cup \{\lambda_i\} \quad (11.55)$$

Step 4: Apply QR decomposition on θ_{\wedge_i}

$$[QR] = Q R D(\theta_{\wedge_i}) \quad (11.56)$$

Step 5: Compute the new residual

$$r_i = r_{i-1} - Q Q^H r_{i-1} \quad (11.57)$$

Step 6: Set $i = i + 1$ if $i < K$ go to step 2

Step 7: Solve $R \hat{h} = Q^H y$ using back substitution to obtain k nonzero coefficients of \hat{h} . OMP takes the measurement matrix θ and the measurement vector y as the inputs. In the first iteration, the column of measurement matrix which has the maximum correlation with the measurement vector is chosen and QR decomposition is performed on it. Now, the residue is updated and used for next iteration. From the second iteration onwards, the column of measurement matrix that has the maximum correlation with the residue is chosen. Finally, after K iterations correct set of columns of θ are obtained. The QR decomposition is performed on θ which gives the final Q and R . Then \hat{h} is obtained by using Q , R , and y as inputs to the back substitution. The QR decomposition is performed using the modified Gram–Schmidt orthogonalization.

Algorithm 2 QR Decomposition

Step 1: Set $[m, n] = \text{size}(\theta_{\wedge_i})$; $Q = \text{zeros}(m, n)$; $R = \text{zeros}(m, n)$;

Step 2: For $j = 1$ to n do

$$v = \theta_{\wedge_i}(:, j) \quad (11.58)$$

for $i = 1$ to $j - 1$

$$R(j, j) = Q(:, j)^H * v \quad (11.59)$$

$$v = v - R(j, j) * Q(:, j) \quad (11.60)$$

end for

$$R(j, j) = \text{norm}(v); Q(:, j) = v/R(j, j) \quad (11.61)$$

end for

Algorithm 3 Back Substitution

Let $b = Q^H y$

Step 1: Set $n = \text{size}(b)$, and $\hat{h} = 0$

Step 2: For $i = n$ to 1 do

$$h(i) = (b(i) - R(i, :) * \hat{h}) / R(i, i) \quad (11.62)$$

The MIMO-OFDM channel estimation is illustrated with 2×2 MIMO-OFDM channel estimation which is decomposed into simultaneous estimation of 4 SISO-OFDM channels \mathbf{h}_{11} , \mathbf{h}_{12} , \mathbf{h}_{21} , and \mathbf{h}_{22} .

The Parameters for the simulated MIMO-OFDM system are:

Number of transmitters (N_T) = 2

Number of receivers (N_R) = 2

Number of fft points (N_{fft}) = 512;

Number of pilots at each transmit antenna(N_P) = 128;

Length of cyclic prefix (N_g) = 64;

Number of channel taps (L) = 64;

Number of nonzero channel coefficients (S) = 16;

Modulation used: 16-QAM

The following MATLAB program 11.8 is written and executed to estimate 2×2 MIMO-OFDM channel. The Eb/No versus normalized mean square error (NMSE) for the 2×2 MIMO-OFDM channel estimation is shown in Fig. 11.13.

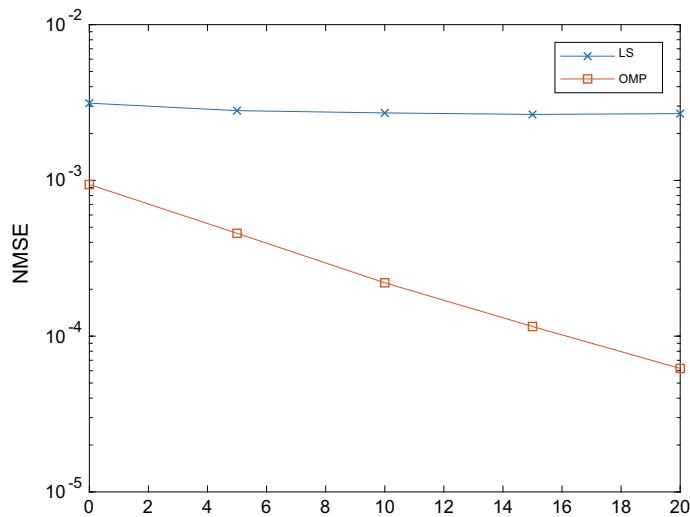


Fig. 11.13 $\frac{E_b}{N_0}$ versus NMSE for 2×2 MIMO-OFDM channel estimation

Program 11.8 MATLAB program for MIMO-OFDM channel estimation

```

close all; clear all;
N_fft=512; N_g=64;%N_fft/8;
N_ofdm=N_fft+N_g; N_sym=100;N_ps=8;
N_p=N_fft/N_ps; N_d=N_fft-N_p; % Pilot spacing, Numbers
% of pilots and data per OFDM symbol
N_bps=4; M=2^N_bps; % Number of bits per (modulated)
symbol
mod_object = modem.qammod('M',M, 'SymbolOrder','gray');
demod_object = modem.qamdemod('M',M, 'SymbolOr-
der','gray');
Es=1; A=sqrt(3/2*(M-1)*Es); % Signal energy& QAM nor-
malization factor
EbN0s = [0:5:20];
for i=1:length(EbN0s)
    EbN0 = EbN0s(i);
    randn('seed',1)
    NMSE_OMPi=0;NMSE_LSi=0;
    for nsym=1:N_sym
        X_p1= 2*(randn(1,N_p)>0)-1;      % Pilot sequence genera-
tion
        msg_int=randi(1,N_fft-N_p,M);      % bit generation
        Data = modulate(mod_object,msg_int)*A; %QAM Mod-
ulated symbols
        ip1 = 0; pilot_loc1 = [];
        for k=1:N_fft
            if mod(k,N_ps)==1
                X1(k) = X_p1(floor(k/N_ps)+1);
                pilot_loc1 = [pilot_loc1 k];
            end
        end
        Demodulated = demod_object(Data);
        % Error calculation
        error = Demodulated - X1;
        NMSE_OMPi = NMSE_OMPi + sum(error.^2);
        NMSE_LSi = NMSE_LSi + sum((Demodulated-X1).^2);
    end
    NMSE_OMP(i)=NMSE_OMPi/N_sym;
    NMSE_LSi(i)=NMSE_LSi/N_sym;
end

```

```

        ip1 = ip1+1;
    else           X1(k) = Data(k-ip1);
    end
end
X_p2= 2*(randn(1,N_p)>0)-1;      % Pilot sequence generation
ip2 = 0;   pilot_loc2 = [];
for k=1:N_fft
    if mod(k,N_ps)==1
        X2(k) = X_p2(floor(k/N_ps)+1);
        pilot_loc2 = [pilot_loc2 k];
        ip2 = ip2+1;
    else           X2(k) = Data(k-ip2);
    end
end
x1 = ifft(X1,N_fft);      % IFFT
xt1 = [x1(N_fft-N_g+1:N_fft) x1]; % Add CP
x2 = ifft(X2,N_fft);      % IFFT
xt2 = [x2(N_fft-N_g+1:N_fft) x2]; % Add CP
L = 64; %Total number of channel taps
K = 7; % non-zero channel taps
T = randperm(L);T = T(1:K);
h11 = zeros(L,1);h11(T) = randn(K,1) + li*randn(K,1);
H11= fft(h11',N_fft); channel_length = length(h11);
y_channel11 = conv(xt1, h11');      % Channel path (convolution)
h12 = zeros(L,1);T = randperm(L);T = T(1:K);
h12(T) = randn(K,1) + li*randn(K,1);
H12= fft(h12',N_fft); channel_length = length(h12); y_channel12
= conv(xt1, h12');      % Channel path (convolution)
h21 = zeros(L,1);T = randperm(L);T = T(1:K);h21(T) =
randn(K,1) + li*randn(K,1);
H21= fft(h21',N_fft); channel_length = length(h21); y_channel21
= conv(xt2, h21');      % Channel path (convolution)
h22 = zeros(L,1);T = randperm(L);T = T(1:K);
h22(T) = randn(K,1) + li*randn(K,1);
H22= fft(h22',N_fft); channel_length = length(h22);
y_channel22 = conv(xt2, h22');      % Channel path (convolution)
yt11= awgn(y_channel11,EbNo,'measured'); %awgn noise added
y11 = yt11(N_g+1:N_ofdm);          % Remove CP
Y11 = fft(y11);                   % FFT
yt12= awgn(y_channel12,EbNo,'measured'); %awgn noise added
y12 = yt12(N_g+1:N_ofdm);          % Remove CP
Y12 = fft(y12);                   % FFT
yt21= awgn(y_channel21,EbNo,'measured'); %awgn noise added
y21 = yt21(N_g+1:N_ofdm);          % Remove CP
Y21 = fft(y21);                   % FFT
yt22= awgn(y_channel22,EbNo,'measured'); %awgn noise added
y22 = yt22(N_g+1:N_ofdm);          % Remove CP
Y22 = fft(y22);                   % FFT
k=1:N_p; Est_LSH11(k) = Y11(pilot_loc1(k))./X_p1(k);
k=1:N_p; Est_LSH12(k) = Y12(pilot_loc1(k))./X_p1(k);
k=1:N_p; Est_LSH21(k) = Y21(pilot_loc2(k))./X_p2(k);
k=1:N_p; Est_LSH22(k) = Y22(pilot_loc2(k))./X_p2(k);
Xp1=diag(X1(pilot_loc1));Xp2=diag(X2(pilot_loc2));

```

```

    yps11=Y11(pilot_loc1); yps12=Y12(pilot_loc1);
    yps21=Y21(pilot_loc2);
    yps22=Y22(pilot_loc2);
    for ii=1:N_p
        ypss11(ii,1)=yps11(ii);    ypss12(ii,1)=yps12(ii);
        ypss21(ii,1)=yps21(ii);    ypss22(ii,1)=yps22(ii);
    end
Est_HLS11 = interpolate(Est_LSH11,pilot_loc1,N_fft,'linear');
h_estLS11 = ifft(Est_HLS11,L);
Est_HLS12 = interpolate(Est_LSH12,pilot_loc1,N_fft,'linear');
h_estLS12 = ifft(Est_HLS12,L);
Est_HLS21 = interpolate(Est_LSH21,pilot_loc2,N_fft,'linear');
h_estLS21 = ifft(Est_HLS21,L);
Est_HLS22 = interpolate(Est_LSH22,pilot_loc2,N_fft,'linear');
h_estLS22 = ifft(Est_HLS22,L);
W =exp(-j*2*pi/N_fft*[0:N_fft-1]'*[0:N_fft-1]);
WW=W(1:N_ps:N_fft,1:L); %sub dft matrix
AA1=diag(X_p1)*WW; AA2=diag(X_p2)*WW;
hompl1=ompes(ypss11,AA1,K); hompl2=ompes(ypss12,AA1,K);
hompl1=ompes(ypss21,AA2,K); hompl2=ompes(ypss22,AA2,K);
H_org=[h11 h12 h21 h22];
H_omp=[ hompl1; hompl2; hompl1; hompl2];
H_LS=[h_estLS11;h_estLS12;h_estLS21;h_estLS22];
NMSE_OMPi= NMSE_OMPi+ (norm(H_org-H_omp')/norm(H_org));
NMSE_LSi= NMSE_LSi+ (norm(H_org-H_LS')/norm(H_org));
end
NMSE_OMP(i)=NMSE_OMPi;
NMSE_LS(i)=NMSE_LSi;
end
NMSE_LS=Nmse_LS/(N_fft*N_sym);
NMSE_OMP=Nmse_OMP/(N_fft*N_sym);
figure, semilogy(EbN0s',NMSE_LS,'-x', EbN0s',NMSE_OMP,'-s')
xlabel ('Eb/NO (dB)'); ylabel ('NMSE'); legend('LS','OMP')

```

The MATLAB function for OMP given in Chap. 2 is used here.

From Fig. 11.13, it is observed that OMP algorithm outperforms the LS estimate 2×2 MIMO-OFDM channel estimation.

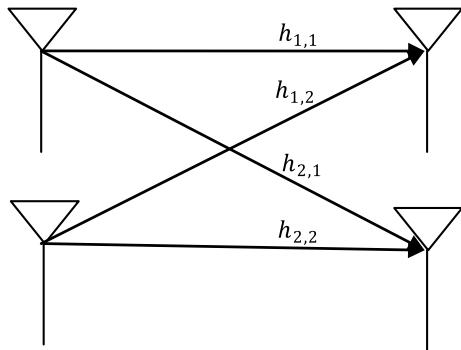
11.7 MIMO Channel Equalization

Consider a wireless communication system in which the transmitter contains N_T antennas and the receiver possesses N_R antennas. Let us consider a 2×2 MIMO channel ($N_T = 2, N_R = 2$) as shown in Fig. 11.14. The received signal for the 2×2 MIMO channel can be expressed as

$$y_j = \sum_{i=1}^2 h_{j,i} s_i + \eta_j \quad (11.63)$$

where $h_{j,i}$ ($i = 1, 2; j = 1, 2$) are independent and identically distributed (i.i.d) complex random variables, representing the channel coefficients from the i th transmitter

Fig. 11.14 A 2×2 MIMO channel



antenna to the j th receiver antenna, x_i are transmitted symbols, η_j are noise samples, and i.i.d complex AWGN variables.

Equation (11.63) can be represented in matrix form as

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} \quad (11.64)$$

equivalently,

$$y = Hs + \eta$$

11.7.1 Zero Forcing (ZF) Equalization

To solve for s , we know that we need to find a matrix W which satisfies $WH = I$. The Zero Forcing (ZF) linear detector for meeting this constraint is given by,

$$W = (H^H H)^{-1} H^H \quad (11.65)$$

This matrix is also known as the pseudo-inverse for a general $m \times n$ matrix. The term,

$$\begin{aligned} H^H H &= \begin{bmatrix} h_{1,1}^* & h_{2,1}^* \\ h_{1,2}^* & h_{2,2}^* \end{bmatrix} \begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \end{bmatrix} \\ &= \begin{bmatrix} |h_{1,1}|^2 + |h_{2,1}|^2 & h_{1,1}^* h_{1,2} + h_{2,1}^* h_{2,2} \\ h_{1,2}^* h_{1,1} + h_{2,2}^* h_{2,1} & |h_{1,2}|^2 + |h_{2,2}|^2 \end{bmatrix} \end{aligned}$$

11.7.2 Minimum Mean Square Error (MMSE) Equalization

The minimum mean square error (MMSE) approach tries to find W which minimizes the criterion $E\left\{\left[W_{y-s}\right]\left[W_{y-s}\right]^H\right\}$. Solving, $W = [H^H H + N_0 I]^{-1} H^H$ when the noise term is zero, the MMSE equalizer reduces to Zero Forcing equalizer.

11.7.2.1 Maximum Likelihood Equalization

ML detection shows the best performance in all the MIMO detection algorithms. It finds the \hat{s} , which minimizes

$$J = |y - H\hat{s}|^2 \quad (11.66)$$

$$J = \left| \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} - \begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \end{bmatrix} \begin{bmatrix} \hat{s}_1 \\ \hat{s}_2 \end{bmatrix} \right|^2 \quad (11.67)$$

If the modulation is BPSK, the possible values of s_1 is $+1$ or -1 . Similarly, s_2 also take values $+1$ or -1 . So, to find the maximum likelihood estimate, we need to find the minimum from the all four combinations of $s_1, s_2, s_2, s_3, s_4 s_1$ and s_2 with J defined for the four combinations as

$$J_{+1,+1} = \left| \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} - \begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \end{bmatrix} \begin{bmatrix} +1 \\ +1 \end{bmatrix} \right|^2 \quad (11.68)$$

$$J_{+1,-1} = \left| \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} - \begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \end{bmatrix} \begin{bmatrix} +1 \\ -1 \end{bmatrix} \right|^2 \quad (11.69)$$

$$J_{-1,+1} = \left| \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} - \begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \end{bmatrix} \begin{bmatrix} -1 \\ +1 \end{bmatrix} \right|^2 \quad (11.70)$$

$$J_{-1,-1} = \left| \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} - \begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \end{bmatrix} \begin{bmatrix} -1 \\ -1 \end{bmatrix} \right|^2 \quad (11.71)$$

In case of 4×4 MIMO, the Eq. (10.44) can be written as

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & h_{1,4} \\ h_{2,1} & h_{2,2} & h_{2,3} & h_{2,4} \\ h_{3,1} & h_{3,2} & h_{3,3} & h_{3,4} \\ h_{4,1} & h_{4,2} & h_{4,3} & h_{4,4} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} + \begin{bmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \\ \eta_4 \end{bmatrix} \quad (11.72)$$

ML detection minimizes

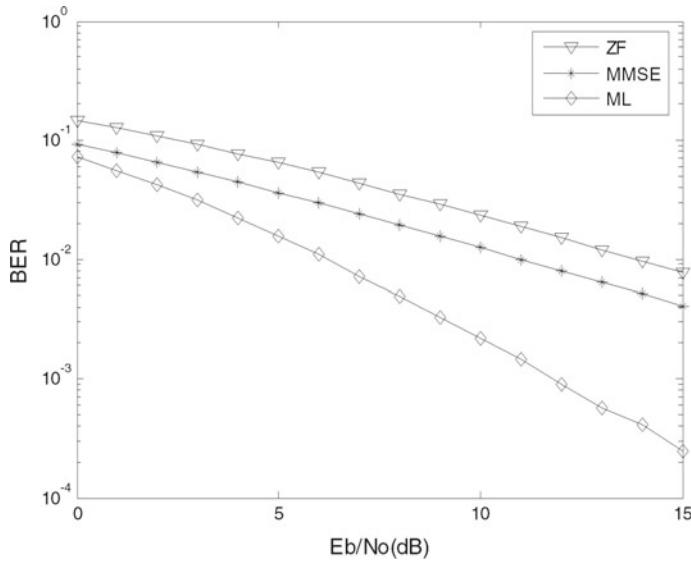


Fig. 11.15 Performance comparison of MIMO channel equalization using ZF, MMSE, and ML

$$J = \left\| \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} - \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & h_{1,4} \\ h_{2,1} & h_{2,2} & h_{2,3} & h_{2,4} \\ h_{3,1} & h_{3,2} & h_{3,3} & h_{3,4} \\ h_{4,1} & h_{4,2} & h_{4,3} & h_{4,4} \end{bmatrix} \begin{bmatrix} \hat{s}_1 \\ \hat{s}_2 \\ \hat{s}_3 \\ \hat{s}_4 \end{bmatrix} \right\|^2 \quad (11.73)$$

with BPSK modulation for maximum likelihood estimate, we need to calculate minimum from all sixteen combinations of s_1, s_2, s_3 , and s_4 .

The performance comparison of MIMO channel equalization using ZF, MMSE, and ML is shown in Fig. 11.15. From Fig. 11.15, it is observed that the performance of MMSE is better than ZF and the performance of ML is better than both the MMSE and ZF.

$$y = Hx + n$$

11.8 Problems

- Find a parallel channel model for a MIMO system, the H matrix of which is given by

$$H = \begin{bmatrix} 0.8 & 0.5 - j0.2 & 0.3 + j0.6 \\ 0.4 - j0.6 & 1.0 - j0.1 & 0.2 - j0.9 \\ 0.5 + j0.3 & 0.5 + j1.5 & 0.6 + j1.2 \end{bmatrix}$$

2. Given the following (3×3) MIMO channel, find the capacity of this channel, with known CSI at the receiver, unknown CSI at the transmitter, SNR = 20 dB and bandwidth equal to 2 kHz. Compare this capacity calculation to that using SVD.

$$H = \begin{bmatrix} 0.8 & 0.5 & 0.3 \\ 0.4 & 1.0 & 0.2 \\ 0.5 & 0.5 & 0.6 \end{bmatrix}$$

3. Consider a MIMO channel with two transmit antennas and one receive antenna. Assume zero mean unit variance AWGN and an average power constraint of one per antenna. If the path gains from the first and second transmit antennas to the receiver antenna are $h_1 = 0.5$ and $h_2 = 0.5 + j1.5$, respectively.
- (a) What is the channel capacity?
 - (b) What is the channel capacity if CSI is known at the transmitter and the average power constraint is 2 over sum of the transmission powers from both the antennas.
4. Assuming total power is 1 W, noise power is equal to 0.1 W and the signal bandwidth is 50 kHz, find the channel capacity and optimal power allocation for MIMO channel, the H matrix of which is given by

$$H = \begin{bmatrix} 0.8 & 0.5 - j0.2 & 0.3 + j0.6 \\ 0.4 - j0.6 & 1.0 - j0.1 & 0.2 - j0.9 \\ 0.5 + j0.3 & 0.5 + j1.5 & 0.6 + j1.2 \end{bmatrix}$$

11.9 MATLAB Exercises

1. Write a MATLAB program for the simulations to estimate the achievable information capacity for BPSK input and QPSK input over a MIMO system, the H matrix of which is given by

$$H = \begin{bmatrix} 0.8 & 0.5 - j0.2 & 0.3 + j0.6 \\ 0.4 - j0.6 & 1.0 - j0.1 & 0.2 - j0.9 \\ 0.5 + j0.3 & 0.5 + j1.5 & 0.6 + j1.2 \end{bmatrix}$$

2. Write a MATLAB program to plot the ergodic channel capacity of a 2×2 MIMO system over ergodic Rayleigh fading channel with transmit correlation matrix $R_t = \begin{bmatrix} 1 & r \\ r & 1 \end{bmatrix}$ and receive correlation matrix $R_r = R_t$ for $r = 0, 0.5, 0.6, 0.8$.

References

1. R.D. Murch, K.B. Letaief, Antenna systems for broadband wireless access. *IEEE Commun. Mag.*, **40**(April), 76–83 (2002)
2. H. Bölskei, A. Paulraj, Space-frequency coded broadband OFDM systems, in *Proceedings of IEEE WCNC*, Chicago, IL, USA, 23–28 Sept 2000, pp. 1–6
3. B. Lu, X. Wang, Space-time code design in OFDM systems, in *Proceedings of the IEEE Global Communication Conference*, Nov 2000, pp. 1000–04
4. F. Tufvesson, T. Maseng, Pilot assisted channel estimation for OFDM in mobile cellular systems. *IEEE VTC'97*, vol. 3, pp. 1639–1643 (May 1997)
5. J. Heiskala, J. Terry, OFDM wireless LANs: a theoretical and practical guide, SAMS (2002)
6. R. van Nee, R. Prasad, *OFDM for Wireless Multimedia Communications* (Artech House Publishers, 2000)
7. H.K. Lau, S.W. Cheung, A pilot symbol aided technique used for digital signals in multipath environments, in *IEEE ICC'94*, vol. 2, pp. 1126–1130 (May 1994)
8. Y.S. Cho, J. Kim, W.Y. Yang, C.G. Kang, *MIMO-OFDM Wireless Communications with MATLAB* (Wiley, 2010)
9. R.B. Ertel, P. Cardieli, K.W. Sowerby, T.S. Rapport, J.H. Reed, Overview of spatial channel models for antenna array communication systems. *IEEE Pers. Commun.* **5**(1), 10–22 (1998)
10. J.P. Kermoal, L. Schumacher, K.I. Pederson, P.E. Modensen, F. Fredriksen, A stochastic MIMO radio channel model with experimental validation. *IEEE J. Sel. Areas Commun.* **20**(1), 1211–1226 (2002)
11. D. Gesbert, H. Bolcskei, D. Gore, A. Paulraj, Outdoor MIMO channels: models and performance prediction. *IEEE Trans. Commun.* **50**(12), 1926–1934 (2002)
12. Sergey Loyka, George Tsoulos, Estimating MIMO system performance using the correlation matrix approach. *IEEE Commun. Lett.* **6**(1), 19–21 (2002)
13. A. van Zelst, J.S. Hammerschmidt, A single coefficient spatial correlation model for multiple-input multiple-output (MIMO) radio channels, in *Proceedings of the URSI 27th General Assembly Maastricht*, Netherlands, 2002, pp. 657–660
14. G.D. Durnin, T.S. Rappaport, Effects of multipath angular spread on the spatial cross-correlation of received voltage envelopes, in *49th IEEE Vehicular Technology Conference (VTC) 1999*, vol. 2, pp. 996–1000
15. X. He, R. Song, W.-P. Zhu, Pilot allocation for sparse channel estimation in MIMO-OFDM systems. *IEEE Trans Circuits Syst—II: Express Br* **60**(9), September (2013)

Chapter 12

Space-Time Coding



In MIMO systems, diversity can be achieved by repetition coding in which different antennas at the transmitter transmit the same information at different time slots. The space-time (ST) coding is more bandwidth efficient coding scheme, which transmits an information symbol block in a different order from each antenna. The diverse copies of the data transmitted are received with multiple receiving antennas. All the copies of the received signal are combined in an optimal way to extract information from each of them. This chapter describes different space-time coding schemes and analyzes their performance in Rayleigh fading.

12.1 Space-Time-Coded MIMO System

A space-time-coded MIMO system with N_T transmit antennas and N_R receive antennas is shown in Fig. 12.1. In this MIMO system, the bit stream is mapped into a symbol stream S_i , $i = 1, \dots, N$. The N symbols are space-time-encoded into s_{ij} , $i = 1, 2, \dots, N_T$, $j = 1, 2, \dots, T$. Where i represents antenna index and j stands for the symbol time index. Thus, s_{ij} , $i = 1, 2, \dots, N_T$, $j = 1, 2, \dots, T$ forms a space-time codeword with the number of symbols $N = N_T \cdot T$.

Space-time codes are categorized as space-time block codes (STBC) and space-time trellis codes (STTC). Sections 12.2 through 12.5 discuss these codes. The performance of STTCs is better than that of STBCs. However, STTCs complexity is more due to the maximum likelihood (ML) decoder in the receiver.

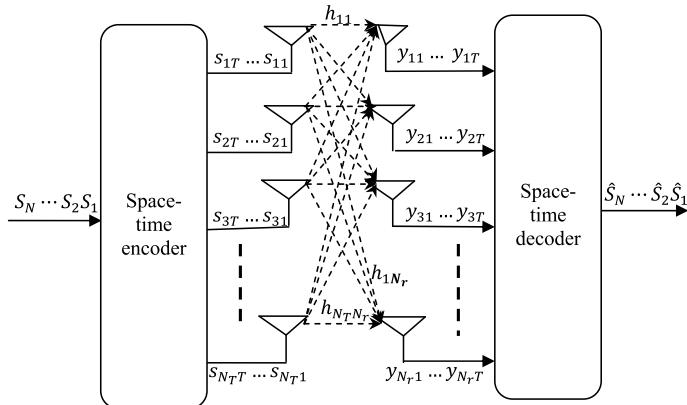


Fig. 12.1 Space-time-coded MIMO system

12.2 Space-Time Block Code (STBC)

An STBC is represented by the following code matrix S in which each row represents one antenna's transmissions over time, and each column represents a time slot.

$$S = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1T} \\ s_{21} & s_{22} & \cdots & s_{2T} \\ \vdots & \vdots & & \vdots \\ s_{N_T 1} & s_{N_T 2} & \cdots & s_{N_T T} \end{bmatrix} \quad (12.1)$$

where s_{ij} is the modulated symbol to be transmitted from antenna i in time slot j . T and N_T represent time slots and transmit antennas, respectively. The code rate of an STBC is defined as how many symbols per time slot it transmits on average. If k symbols are transmitted over T time slots, the code rate of STBC is

$$r = \frac{k}{T} \quad (12.2)$$

The matrix of STBC is to be designed so that it achieves highest possible diversity of $N_T N_R$ and highest possible code rate with minimum complexity of the decoder.

12.2.1 Rate Limit

It is proved in [1] that a code with N_T transmit antennas will yield the highest rate given by

$$r_{\max} = \frac{a_0 + 1}{2a_0} \quad (12.3)$$

where $N_T = 2a_0$ or $N_T = 2a_0 - 1$.

12.2.2 Orthogonality

STBC is to be designed such that any pair of columns taken from the code matrix is orthogonal in order to make the decoding process at the receiver to be simple, linear, and optimal. However, a code that satisfies this criterion must sacrifice a part of its rate.

12.2.3 Diversity Criterion

Orthogonal STBCs can be shown to achieve the maximum diversity allowed by diversity criterion derived in [2]. Consider a codeword

$$c = c_{11}c_{21} \dots c_{N_T 1}c_{12}c_{22} \dots c_{N_T 2} \dots c_{1T}c_{2T} \dots c_{N_T T} \quad (12.4)$$

and let the corresponding erroneously decoded codeword

$$\tilde{c} = \tilde{c}_{11}\tilde{c}_{21} \dots \tilde{c}_{N_T 1}\tilde{c}_{12}\tilde{c}_{22} \dots \tilde{c}_{N_T 2} \dots \tilde{c}_{1T}\tilde{c}_{2T} \dots \tilde{c}_{N_T T} \quad (12.5)$$

Then, the $N_T \times T$ difference matrix $E(c, \tilde{c})$ can be defined as

$$E(c, \tilde{c}) = \begin{vmatrix} \tilde{c}_{11} - c_{11} & \tilde{c}_{12} - c_{12} & \dots & \tilde{c}_{1T} - c_{1T} \\ \tilde{c}_{21} - c_{21} & \tilde{c}_{22} - c_{22} & \dots & \tilde{c}_{2T} - c_{1T} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{c}_{N_T 1} - c_{N_T 1} & \tilde{c}_{N_T 2} - c_{N_T 2} & \dots & \tilde{c}_{N_T T} - c_{N_T T} \end{vmatrix} \quad (12.6)$$

Rank and Determinant Criteria

Let ρ ($\rho \leq N_T$) be the rank of difference matrix $E(c, \tilde{c})$. The $E(c, \tilde{c})$ should be full-rank matrix for any pair of distinct code words c and \tilde{c} to yield maximum possible diversity order of $N_T N_R$. Instead, if $E(c, \tilde{c})$ has minimum rank ρ over the set distinct code words pairs, then diversity order is ρN_R [2]. Consider the following distance matrix

$$A(c, \tilde{c}) = E(c, \tilde{c})E^*(c, \tilde{c}) \quad (12.7)$$

where $E^*(c, \tilde{c})$ denotes the transpose conjugate of $E(c, \tilde{c})$.

The determinant criterion states that the minimum determinant of $A(c^i, c^j) = E(c^i, c^j)^H E(c^i, c^j)$ among all $i \neq j$ should be large to achieve high coding gains.

Trace Criterion

A good design criterion is to maximize the minimum distance $\|E(c, \tilde{c})\|_F$ for all $i \neq j$. This is called the trace criterion because $\|E(c, \tilde{c})\|_F^2 = \text{Tr}[A(c, \tilde{c})]$. The metric $\|E(c, \tilde{c})\|_F$ provides all the good properties of a distance measure.

12.2.4 Performance Criteria

The rank of A is ρ , the kernel of A has a minimum dimension $N_{\text{Tx}} - \rho$, and exactly $N_{\text{Tx}} - \rho$ eigenvalues of A are zero. The nonzero eigenvalues of A can be denoted by $\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_\rho$. Assuming perfect channel state information (CSI), the probability of transmitting c and deciding in favor of \tilde{c} at the decoder is given by [3, 4]

$$P(c \rightarrow \tilde{c} | h_{ij}, i = 1, 2, \dots, N_{\text{Tx}}; j = 1, 2, \dots, N_{\text{Rx}}) \leq \exp\left(-d^2(c, \tilde{c}) \frac{E_s}{4N_0}\right) \quad (12.8)$$

where $\frac{N_0}{2}$ is the noise variance per dimension and

$$d^2(c, \tilde{c}) = \sum_{j=1}^{N_{\text{Rx}}} \sum_{t=1}^T \left| \sum_{i=1}^{N_{\text{Tx}}} h_{ij} (c_t^i - \tilde{c}_t^i) \right|^2 \quad (12.9)$$

is the Euclidean distance.

It follows from [5] that the pairwise error bound is given by

$$P(c \rightarrow \tilde{c}) \leq \left(\prod_{i=1}^{\rho} \lambda_i \right)^{-N_{\text{Rx}}} \left(\frac{E_s}{4N_0} \right)^{-\rho N_{\text{Rx}}} \quad (12.10)$$

To achieve the best performance for a given system, the rank and determinant criteria should be satisfied [4].

12.2.5 Decoding STBCs

One particular attractive feature of orthogonal STBCs is that maximum likelihood decoding can be achieved at the receiver with only linear processing. In order to consider a decoding method, a model of the wireless communications system is needed.

At time t , the signal y_{jt} received at antenna j is:

$$y_{jt} = \sum_{i=1}^{N_T} h_{ij} s_{it} + \eta_{jt} \quad (12.11)$$

where h_{ij} is the path gain from transmit antenna i to receive antenna j , s_{it} is the signal transmitted by transmit antenna i , and η_{jt} is the additive white Gaussian noise (AWGN).

The decision variables are formed by the maximum likelihood detection rule [3]

$$Y_i = \sum_{t=1}^{N_T} \sum_{j=1}^{N_R} y_{jt} h_{\epsilon_k(i),j} \delta_k(i) \quad (12.12)$$

where $\delta_k(i)$ is the sign of s_i in the k th row of the coding matrix, $\epsilon_k(p) = q$ denotes that s_p is (up to a sign difference), the (k, q) element of the coding matrix, for $i = 1, 2, \dots, n_T$ and then decide on constellation symbol s_i that satisfies

$$s_i = \arg \min_{s \in A} \left(|Y_i - s|^2 + \left(-1 + \sum_{k,l} |h_{kl}|^2 \right) |s|^2 \right) \quad (12.13)$$

with A the constellation alphabet. Despite its appearance, this is a simple, linear decoding scheme that provides maximal diversity.

12.3 Alamouti Code

The very first and well-known STBC is the Alamouti code [5]. In the Alamouti encoder, two consecutive symbols s_1 and s_2 are encoded with the following space-time codeword matrix:

$$S = \begin{bmatrix} s_1 & -s_2^* \\ s_2 & s_1^* \end{bmatrix} \quad (12.14)$$

This indicates that, during the first time slot, signal s_1 and s_2 are transmitted from antenna 1 and antenna 2, respectively. During the next time slot, antenna 1 and antenna 2 transmit $-s_2^*$ and s_1^* , respectively. It is only STBC in which maximum diversity can be achieved without sacrificing its data rate because Alamouti code has rate 1 as it takes two time slots to transmit two symbols.

12.3.1 2-Transmit, 1-Receive Alamouti STBC Coding

For Alamouti scheme with two transmit and one receive antennas is shown in Fig. 12.2, if y_1 and y_2 denote the signals received at first time slot and second time slot, respectively. We have

$$\begin{bmatrix} y_1 & y_2 \end{bmatrix} = \begin{bmatrix} h_1 & h_2 \end{bmatrix} \begin{bmatrix} s_1 & -s_2^* \\ s_2 & s_1^* \end{bmatrix} + \begin{bmatrix} \eta_1 & \eta_2 \end{bmatrix}$$

$$= \begin{bmatrix} h_1 s_1 + h_2 s_2 + \eta_1 & -h_1 s_2^* + h_2 s_1^* + \eta_2 \end{bmatrix} \quad (12.15)$$

s_1, s_2 are the transmitted symbols, h_1 is the channel from first transmit antenna to receive antenna, h_2 is the channel from second transmit antenna to receive antenna, and η_1, η_2 are the noise at time slot 1 and time slot 2.

The combiner generates [2]

$$\tilde{s}_1 = h_1^* y_1 + h_2^* y_2 \quad (12.16)$$

and

$$\tilde{s}_2 = h_2^* y_1 - h_1^* y_2 \quad (12.17)$$

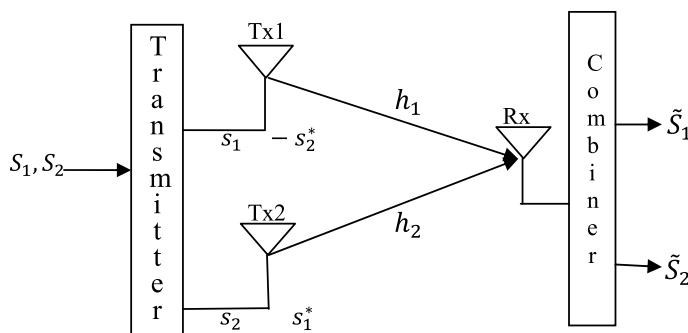


Fig. 12.2 Alamouti scheme with two transmit and one receive antennas

To decode, the ML decoder minimizes the following decision metric (for decoding) s_1 and s_2 , respectively [3].

$$|\tilde{s}_1 - s_1|^2 + \xi |s_1|^2 \quad (12.18)$$

$$|\tilde{s}_2 - s_2|^2 + \xi |s_2|^2 \quad (12.19)$$

where

$$\xi = \left(-1 + \sum_{i=1}^{N_t} |h_i|^2 \right) \quad (12.20)$$

BER with Alamouti (2×1) STBC

From Eq. (2.44), BER for a 2-branch MRC (i.e., with one transmitting and two receiving antennas) with BPSK modulation can be expressed as

$$\text{BER}_{\text{MRC}(1 \times 2)} = p_{\text{MRC}}^2 [1 + 2(1 - p_{\text{MRC}})] \quad (12.21)$$

where

$$p_{\text{MRC}} = \frac{1}{2} - \frac{1}{2} \left(1 + \frac{1}{E_b/N_0} \right)^{-1/2} \quad (12.22)$$

Then, the BER for the Alamouti 2-transmit, 1-receive antenna STBC case with BPSK modulation can be written as

$$\text{BER}_{\text{Alamouti } (2 \times 1)} = p_{\text{Alamouti}}^2 [1 + 2(1 - p_{\text{Alamouti}})] \quad (12.23)$$

where

$$p_{\text{Alamouti}} = \frac{1}{2} - \frac{1}{2} \left(1 + \frac{2}{E_b/N_0} \right)^{-1/2} \quad (12.24)$$

It can be easily shown [5] that the performance of the Alamouti scheme with two transmitters and a single receiver is identical to that of the two-branch MRC provided that each transmit antenna in the Alamouti scheme radiates the same energy as the single transmit antenna for MRC.

12.3.2 2-Transmit, 2-Receive Alamouti STBC Coding

For Alamouti Scheme with two transmit and two receive antennas is shown in Fig. 12.3, if $y_{11}, y_{12}, y_{21}, y_{22}$ denote the signals received by antenna 1 at first time

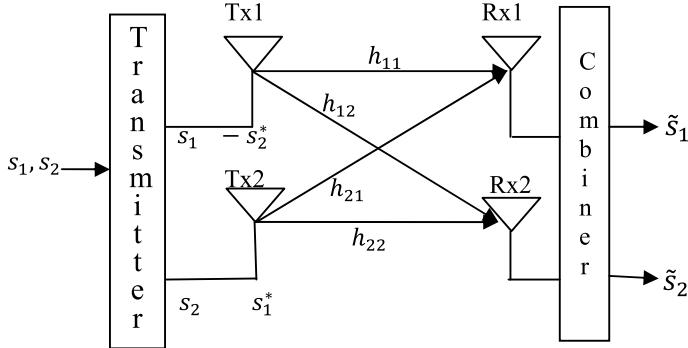


Fig. 12.3 Alamouti scheme with two transmit and two receive antennas

slot, by antenna 1 at second time slot, by antenna 2 at first time slot, and by antenna 2 at second time slot, respectively. We have

$$\begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} s_1 & -s_2^* \\ s_2 & s_1^* \end{bmatrix} + \begin{bmatrix} \eta_{11} & \eta_{12} \\ \eta_{21} & \eta_{22} \end{bmatrix} \quad (12.25)$$

$$= \begin{bmatrix} h_{11}s_1 + h_{12}s_2 + \eta_{11} & -h_{11}s_2^* + h_{12}s_1^* + \eta_{12} \\ h_{21}s_1 + h_{22}s_2 + \eta_{21} & -h_{21}s_2^* + h_{22}s_1^* + \eta_{22} \end{bmatrix} \quad (12.26)$$

h_{ij} is the channel from i th transmit antenna to j th receive antenna, s_1, s_2 are the transmitted symbols, $\begin{bmatrix} \eta_{11} \\ \eta_{12} \end{bmatrix}$ are the noise at time slot 1 on receive antenna 1 and 2, respectively, and $\begin{bmatrix} \eta_{21} \\ \eta_{22} \end{bmatrix}$ are the noise at time slot 2 on receive antenna 1 and 2, respectively.

The combiner generates [2]

$$\tilde{s}_1 = h_{11}^*y_{11} + h_{12}y_{12}^* + h_{21}^*y_{21} + h_{22}y_{22}^* \quad (12.27)$$

and

$$\tilde{s}_2 = h_{12}^*y_{11} - h_{11}y_{12}^* + h_{22}^*y_{21} - h_{21}y_{22}^* \quad (12.28)$$

To decode, the ML decoder minimizes the following decision metric (for decoding) s_1 and s_2 , respectively [6].

$$|\tilde{s}_1 - s_1|^2 + \xi|s_1|^2 \quad (12.29)$$

$$|\tilde{s}_2 - s_2|^2 + \xi|s_2|^2 \quad (12.30)$$

where

$$\xi = \left(-1 + \sum_{i=1}^{N_r} \sum_{j=1}^{N_t} |h_{i,j}|^2 \right) \quad (12.31)$$

BER with Alamouti (2×2) STBC

From Eq. (2.44), BER for 4-branch MRC (i.e., with one transmitting and four receiving antennas) with BPSK modulation can be expressed as

$$\text{BER}_{\text{MRC}(1 \times 4)} = p_{\text{MRC}}^4 [1 + 4(1 - p_{\text{MRC}}) + 10(1 - p_{\text{MRC}})^2 + 20(1 - p_{\text{MRC}})^3] \quad (12.32)$$

BER for Alamouti (2×2) STBC case with BPSK modulation can be written as

$$\text{BER}_{\text{Alamouti}(2 \times 2)} = p_{\text{Alamouti}}^4 [1 + 4(1 - p_{\text{Alamouti}}) + 10(1 - p_{\text{Alamouti}})^2 + 20(1 - p_{\text{Alamouti}})^3] \quad (12.33)$$

Example 12.1 Consider a Alamouti (2×1) STBC with $h_1 = 2 + j3$ and $h_2 = 4 + j5$. Verify for the orthogonality and detect s_1 and s_2 .

Solution

$$\begin{aligned} y_1 &= [2 + j3 \ 4 + j5] \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \eta_1 \\ y_2 &= [2 + j3 \ 4 + j5] \begin{bmatrix} -s_2^* \\ s_1^* \end{bmatrix} + \eta_2 \\ y_2^* &= [2 - j3 \ 4 - j5] \begin{bmatrix} -s_2 \\ s_1 \end{bmatrix} + \eta_2^* \\ &= [-2 + j3 \ 4 - j5] \begin{bmatrix} s_2 \\ s_1 \end{bmatrix} + \eta_2^* \\ &= [4 - j5 \ -2 + j3] \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \eta_2^* \\ \begin{bmatrix} y_1 \\ y_2^* \end{bmatrix} &= \begin{bmatrix} 2 + j3 & 4 + j5 \\ 4 - j5 & -2 + j3 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \begin{bmatrix} \eta_1 \\ \eta_2^* \end{bmatrix} \end{aligned}$$

Stacking both the symbols convert the Alamouti coded system into 2×2 MIMO with the channel matrix

$$\begin{bmatrix} 2 + j3 & 4 + j5 \\ 4 - j5 & -2 + j3 \end{bmatrix}$$

The columns of the channel matrix c_1 and c_2 are given by

$$c_1 = \begin{bmatrix} 2 + j3 \\ 4 - j5 \end{bmatrix}; c_2 = \begin{bmatrix} 4 + j5 \\ -2 + j3 \end{bmatrix}$$

For orthogonality $c_1^H c_2 = 0$

$$\begin{aligned} [2 + j3 \ 4 - j5]^* \begin{bmatrix} 4 + j5 \\ -2 + j3 \end{bmatrix} &= [2 - j3 \ 4 + j5] \begin{bmatrix} 4 + j5 \\ -2 + j3 \end{bmatrix} \\ &= (2 - j3)(4 + j5) + (4 + j5)(-2 + j3) = 0 \end{aligned}$$

The beam former weight to detect s_1 can be computed as

$$\begin{aligned} w_1 &= \frac{c_1}{\|c_1\|} = \frac{1}{\sqrt{54}} \begin{bmatrix} 2 + j3 \\ 4 - j5 \end{bmatrix} \\ s_1 &= w_1^H \begin{bmatrix} y_1 \\ y_2^* \end{bmatrix} = \frac{1}{\sqrt{54}} [2 - j3 \ 4 + j5] \begin{bmatrix} y_1 \\ y_2^* \end{bmatrix} \end{aligned}$$

The beam former weight to detect s_2 can be computed as

$$\begin{aligned} w_2 &= \frac{c_2}{\|c_2\|} = \frac{1}{\sqrt{54}} \begin{bmatrix} 4 + j5 \\ -2 + j3 \end{bmatrix} \\ s_2 &= w_2^H \begin{bmatrix} y_1 \\ y_2^* \end{bmatrix} = \frac{1}{\sqrt{54}} [4 - j5 \ -2 + j3] \begin{bmatrix} y_1 \\ y_2^* \end{bmatrix} \end{aligned}$$

Example 12.2 Show that the Alamouti code provides full diversity

Solution For considered symbols s_1 and s_2 , the transmitted code word matrix is

$$S = \begin{bmatrix} s_1 & -s_2^* \\ s_2 & s_1^* \end{bmatrix}$$

Now, consider a different pair of symbols $(\check{s}_1, \check{s}_2)$ and the corresponding code word matrix is

$$\check{S} = \begin{bmatrix} \check{s}_1 & -\check{s}_2^* \\ \check{s}_2 & \check{s}_1^* \end{bmatrix}$$

Then, the difference matrix $E(S, \check{S})$ is given by

$$E(S, \check{S}) = \begin{bmatrix} s_1 - \check{s}_1 & \check{s}_2^* - s_2^* \\ s_2 - \check{s}_2 & s_1^* - \check{s}_1^* \end{bmatrix}$$

The determinant of the difference matrix $E(S, \check{S})$ is given by

$$\det[E(S, \check{S})] = |s_1 - \check{s}_1|^2 + |s_2 - \check{s}_2|^2$$

The determinant of the difference matrix is zero if and only if $s_1 = \check{s}_1$ and $s_2 = \check{s}_2$.

Hence, $E(S, \check{S})$ is always full rank when $S \neq \check{S}$ and the determinant criteria is satisfied. Thus, the Alamouti code provides diversity of 4 for 2 receive antennas, and therefore, Alamouti code is a full diversity code.

12.3.3 Theoretical BER Performance of BPSK Alamouti Codes Using MATLAB

The following MATLAB program illustrates the BER performance of uncoded coherent BPSK for MRC and Alamouti STBC in Rayleigh fading channel.

Program 12.1: BER performance of MRC and Alamouti coding

```
% BER performance of Alamouti(2x1),Alamouti(2x2) using BPSK modulation
clear; clc;
Eb_N0_dB=[0:2:20];
for i=1:length(Eb_N0_dB)
    EbN0=Eb_N0_dB(i); sigma=sqrt(0.5/(10^(EbN0/10)));
    EbN0Lin = 10.^(EbN0/10); p = 1/2 - 1/2*(1+1./EbN0Lin).^(1/2);
    p1 = 1/2 - 1/2*(1+2./EbN0Lin).^(1/2);
    BER1(i) = 0.5.*((1-p).^(1/2));
    BER2(i) = p1.^2.*((1-p).^(1/2));
    BER3(i) = p.^2.*((1-p).^(1/2));
    BER4(i) = p1.^4.*((1-p).^(1/2))+(10*(1-p).^(1/2));
    BER5(i) = p.^4.*((1-p).^(1/2))+(10*(1-p).^(1/2));
end % End of FOR loop for Eb_N0_dB
semilogy(Eb_N0_dB,BER1,'+'),hold on;
semilogy(Eb_N0_dB,BER2,'*'),semilogy(Eb_N0_dB,BER3,'o');
semilogy(Eb_N0_dB,BER4,'*'),semilogy(Eb_N0_dB,BER5,'x');
axis([Eb_N0_dB([1 end]) 1e-6 1e0]);
grid on; xlabel('Eb/No(dB)'), ylabel('BER');
legend('SISO (no diversity)', 'Alamouti(2x1)', 'MRC(1x2)', 'Alamouti(2x2)', 'MRC(1x4)');
```

BER performance is obtained by using Program 12.1 is shown in Fig. 12.4.

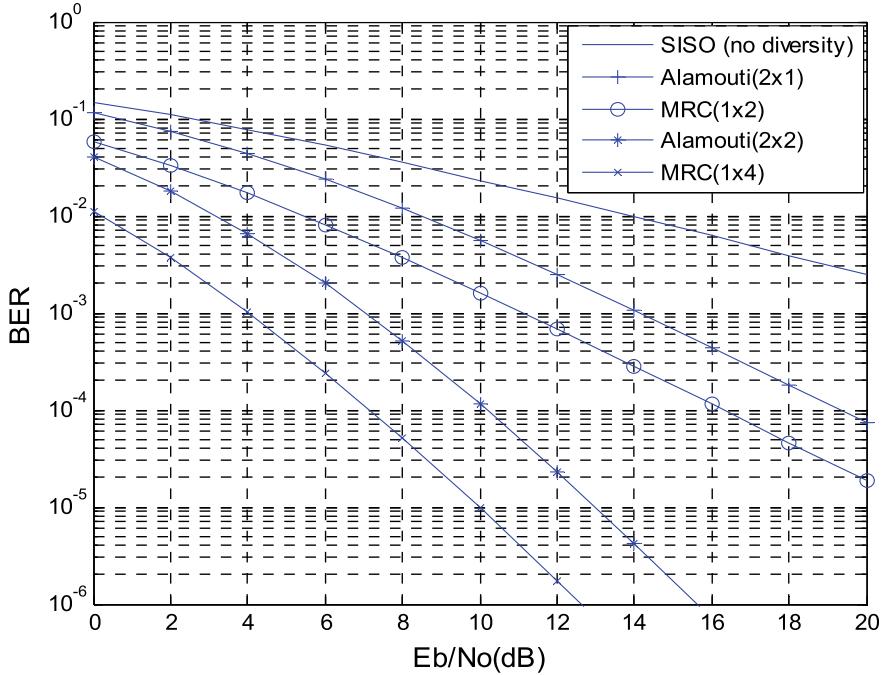


Fig. 12.4 BER performance comparison of coherent BPSK with MRC and Alamouti STBC

From Fig. 12.4, the performance of Alamouti 2×1 STBC and Alamouti 2×2 STBC is 3 dB worse as compared to 1×2 MRC and 1×4 MRC, respectively. The 3 dB penalty is due to the assumption that each transmit antenna in the Alamouti STBC scheme radiates half the energy in order to ensure the same total radiated power as with one transmit antenna of MRC. If each transmit antenna in Alamouti scheme radiates the same energy as the single transmit antenna for MRC, then the performance of Alamouti scheme and MRC are identical.

12.4 Higher-Order STBCs

The Alamouti scheme discussed in Sect. 12.3 is part of a general class of STBCs known as Orthogonal Space-Time Block Codes (OSTBCs) [2]. It is proved in [6, 7] that no code for more than 2 transmit antennas can achieve full rate. This section briefly discusses the full diversity complex orthogonal codes for $N_T > 2$.

3 Transmit Antennas

The full diversity, rate 1/2 code for $N_T = 3$ is given by [6, 7]: This code transmits 4 symbols every 8 time intervals and therefore has rate 1/2.

$$G_3 = \begin{bmatrix} S_1 - S_2 - S_3 - S_4 & S_1^* - S_2^* - S_3^* - S_4^* \\ S_2 & S_1 & S_4 - S_3 & S_2^* & S_1^* & S_4^* - S_3^* \\ S_3 - S_4 & S_1 & S_2 & S_3^* - S_4^* & S_1^* & S_2^* \end{bmatrix} \quad (12.34)$$

4 Transmit Antennas

In the case of 4 transmit antennas, the rate 1/2 code block is given by [6, 7]: where, similar to Eq. (12.34), has rate 1/2 as 4 symbols are transmitted in 8 time intervals

$$G_4 = \begin{bmatrix} s_1 - s_2 - s_3 - s_4 & s_1^* - s_2^* - s_3^* - s_4^* \\ s_2 & s_1 & s_4 - s_3 & s_2^* & s_1^* & s_4^* - s_3^* \\ s_3 - s_4 & s_1 & s_2 & s_3^* - s_4^* & s_1^* & s_2^* \\ s_4 & s_3 - s_2 & s_1 & s_4^* & s_3^* - s_2^* & s_1^* \end{bmatrix} \quad (12.35)$$

12.4.1 3-Transmit, 4-Receive STBC Coding

A STBC scheme with three transmit and four receive antennas is shown in Fig. 12.5. If $y_{11}, y_{12}, \dots, y_{18}$; $y_{21}, y_{22}, \dots, y_{28}$; $y_{31}, y_{32}, \dots, y_{38}$; and $y_{41}, y_{42}, \dots, y_{48}$ denote the signals received by antenna 1; antenna 2; antenna 3 and antenna 4 at time slots 1, 2,..., 8, respectively, and h_{ij} ($i = 1, 2, 3$; $j = 1, 2, 3, 4$) are path gains from antenna i to antenna j . We have

$$\begin{bmatrix} y_{11} & y_{12} & y_{13} & y_{14} & y_{15} & y_{16} & y_{17} & y_{18} \\ y_{21} & y_{22} & y_{23} & y_{24} & y_{25} & y_{26} & y_{27} & y_{28} \\ y_{31} & y_{32} & y_{33} & y_{34} & y_{35} & y_{36} & y_{37} & y_{38} \\ y_{41} & y_{42} & y_{43} & y_{44} & y_{45} & y_{46} & y_{47} & y_{48} \end{bmatrix} = \begin{bmatrix} h_{11} & h_{21} & h_{31} \\ h_{12} & h_{22} & h_{32} \\ h_{13} & h_{23} & h_{33} \\ h_{14} & h_{24} & h_{34} \end{bmatrix} \begin{bmatrix} S_1 - S_2 - S_3 - S_4 & S_1^* - S_2^* - S_3^* - S_4^* \\ S_2 & S_1 & S_4 - S_3 & S_2^* & S_1^* & S_4^* - S_3^* \\ S_3 - S_4 & S_1 & S_2 & S_3^* - S_4^* & S_1^* & S_2^* \end{bmatrix} + \begin{bmatrix} \eta_{11} & \eta_{12} & \eta_{13} & \eta_{14} & \eta_{15} & \eta_{16} & \eta_{17} & \eta_{18} \\ \eta_{21} & \eta_{22} & \eta_{23} & \eta_{24} & \eta_{25} & \eta_{26} & \eta_{27} & \eta_{28} \\ \eta_{31} & \eta_{32} & \eta_{33} & \eta_{34} & \eta_{35} & \eta_{36} & \eta_{37} & \eta_{38} \\ \eta_{41} & \eta_{42} & \eta_{43} & \eta_{44} & \eta_{45} & \eta_{46} & \eta_{47} & \eta_{48} \end{bmatrix} \quad (12.36)$$

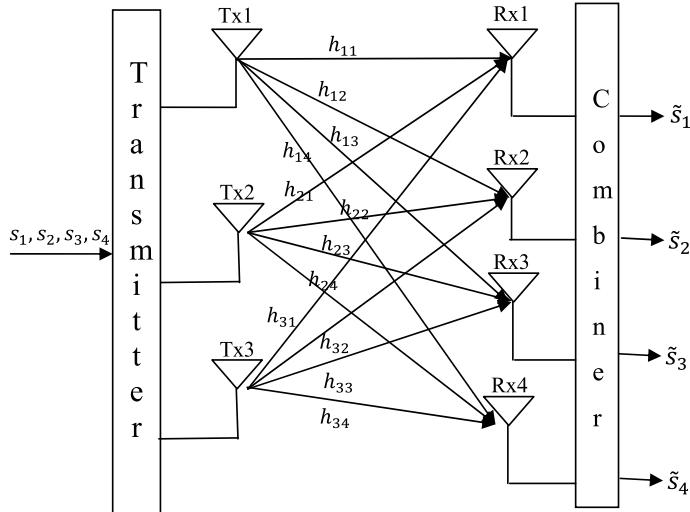


Fig. 12.5 STBC scheme with three transmit and four receive antennas

The Decoding Algorithm

Maximum likelihood decoding of any space-time block code can be achieved using only linear processing at the receiver, and we illustrate this by example. The space-time block code G_3 has s_1, s_2, s_3 and s_4 and their conjugates. These symbols are transmitted simultaneously from antennas one, two, and three, respectively. Then, maximum likelihood detection amounts to minimizing the decision metric

$$\sum_{j=1}^m \left(|y_{1j} - h_{1j}s_1 - h_{2j}s_2 - h_{3j}s_3|^2 + |y_{2j} + h_{1j}s_2 - h_{2j}s_1 + h_{3j}s_4|^2 + |y_{3j} + h_{1j}s_3 - h_{2j}s_4 - h_{3j}s_1|^2 + |y_{4j} + h_{1j}s_4 + h_{2j}s_3 - h_{3j}s_2|^2 + |y_{5j} - h_{1j}s_1^* - h_{2j}s_2^* - h_{3j}s_3^*|^2 + |y_{6j} + h_{1j}s_2^* - h_{2j}s_1^* + h_{3j}s_4^*|^2 + |y_{7j} + h_{1j}s_3^* - h_{2j}s_4^* - h_{3j}s_1^*|^2 + |y_{8j} + h_{1j}s_4^* + h_{2j}s_3^* - h_{3j}s_2^*|^2 \right) \quad (12.37)$$

over all possible values of s_1, s_2, s_3 and s_4 . Note that due to the quasi-static nature of the channel, the path gains are constant over transmissions. The minimizing values are the receiver estimates of s_1, s_2, s_3 and s_4 , respectively. We expand the above metric and delete the terms that are independent of the codeword and observe that the above minimization is equivalent to minimizing

$$\begin{aligned}
& - \sum_{j=1}^m [y_{1j}h_{1j}^*s_1^* + y_{1j}h_{2j}^*s_2^* + y_{1j}h_{3j}^*s_3^* - y_{2j}h_{1j}^*s_2^* + y_{2j}h_{2j}^*s_1^* - y_{2j}h_{3j}^*s_4^* \\
& \quad - y_{3j}h_{1j}^*s_3^* + y_{3j}h_{2j}^*s_4^* + y_{3j}h_{3j}^*s_1^* - y_{4j}h_{1j}^*s_4^* - y_{4j}h_{2j}^*s_3^* + y_{4j}h_{3j}^*s_2^* \\
& \quad + (y_{5j})^*h_{1j}s_1 + (y_{5j})^*h_{3j}s_3 - (y_{6j})^*h_{1j}s_2 + (y_{6j})^*h_{2j}s_1 - (y_{6j})^*h_{3j}s_4 \\
& \quad - (y_{7j})^*h_{1j}s_3 + (y_{7j})^*h_{2j}s_4 + (y_{7j})^*h_{3j}s_1 - (y_{8j})^*h_{1j}s_4 - (y_{8j})^*h_{2j}s_3 \\
& \quad + (y_{8j})^*h_{3j}s_2 + (|s_1|^2 + |s_2|^2 + |s_3|^2 + |s_4|^2) \sum_{j=1}^m \sum_{i=1}^3 |h_{ij}|^2] \quad (12.38)
\end{aligned}$$

The above metric decomposes into four parts, the function of s_1 is

$$\begin{aligned}
& - \sum_{j=1}^m (y_{1j}h_{1j}^*s_1^* + y_{2j}h_{2j}^*s_1^* + y_{3j}h_{3j}^*s_1^* + (y_{5j})^*h_{1j}s_1 + (y_{6j})^*h_{2j}s_1 \\
& \quad + (y_{7j})^*h_{3j}s_1) + (|s_1|^2) \sum_{j=1}^m \sum_{i=1}^3 |\alpha_{i,j}|^2 \quad (12.39)
\end{aligned}$$

The function of s_2 is

$$\begin{aligned}
& - \sum_{j=1}^m (y_{1j}h_{2j}^*s_2^* - y_{2j}h_{1j}^*s_2^* + y_{4j}h_{3j}^*s_2^* + (y_{5j})^*\alpha_{2j}s_2 - (y_{6j})^*\alpha_{1j}s_2 \\
& \quad + (y_{8j})^*\alpha_{3j}s_2) + (|s_2|^2) \sum_{j=1}^m \sum_{i=1}^3 |h_{ij}|^2 \quad (12.40)
\end{aligned}$$

The function of s_3 is

$$\begin{aligned}
& - \sum_{j=1}^m (y_{1j}h_{3j}^*s_3^* - y_{3j}h_{1j}^*s_3^* - y_{4j}h_{2j}^*s_3^* + (y_{5j})^*h_{3j}s_3 - (y_{7j})^*h_{1j}s_3 \\
& \quad - (y_{8j})^*h_{2j}s_3) + (|s_3|^2) \sum_{j=1}^m \sum_{i=1}^3 |h_{ij}|^2 \quad (12.41)
\end{aligned}$$

The function of s_4 is

$$\begin{aligned} & - \sum_{j=1}^m (y_{2j} h_{3j}^* s_4^* + y_{3j} h_{2j}^* s_4^* - y_{4j} h_{1j}^* s_4^* + (y_{6j})^* h_{3j} s_4 + (y_{7j})^* h_{2j} s_4 \\ & - (y_{8j})^* h_{1j} s_4) + (|s_4|^2) \sum_{j=1}^m \sum_{i=1}^3 |h_{ij}|^2 \end{aligned} \quad (12.42)$$

Thus, the minimization of (12.38) is equivalent to minimizing these four parts separately. This in turn is equivalent to minimizing, the decision metric for detecting s_1 is

$$\begin{aligned} & \left| \sum_{j=1}^m (y_{1j} h_{1j}^* + y_{2j} h_{2j}^* + y_{3j} h_{3j}^* + (y_{5j})^* h_{1j} + (y_{6j})^* h_{2j} + (y_{7j})^* h_{3j}) - s_1 \right|^2 \\ & + \left(-1 + 2 \sum_{j=1}^m \sum_{i=1}^3 |h_{ij}|^2 \right) |s_1|^2 \end{aligned} \quad (12.43)$$

The decision metric for detecting s_2 is

$$\begin{aligned} & \left| \sum_{j=1}^m (y_{1j} h_{2j}^* - y_{2j} h_{1j}^* + y_{4j} h_{3j}^* + (y_{5j})^* h_{2j} - (y_{6j})^* h_{1j} + (y_{8j})^* h_{3j}) - s_2 \right|^2 \\ & + \left(-1 + 2 \sum_{j=1}^m \sum_{i=1}^3 |h_{ij}|^2 \right) |s_2|^2 \end{aligned} \quad (12.44)$$

The decision metric for detecting s_3 is

$$\begin{aligned} & \left| \sum_{j=1}^m (y_{1j} h_{3j}^* - y_{3j} h_{1j}^* - y_{4j} h_{2j}^* + (y_{5j})^* h_{3j} - (y_{7j})^* h_{1j} - (y_{8j})^* h_{2j}) - s_3 \right|^2 \\ & + \left(-1 + 2 \sum_{j=1}^m \sum_{i=1}^3 |h_{ij}|^2 \right) |s_3|^2 \end{aligned} \quad (12.45)$$

The decision metric for detecting s_4 is

$$\left| \sum_{j=1}^m (-y_{2j}h_{3j}^* + y_{3j}h_{2j}^* - y_{4j}h_{1j}^* - (y_{6j})^*h_{3j} + (y_{7j})^*h_{2j} - (y_{8j})^*h_{1j}) - s_4 \right|^2 + \left(-1 + 2 \sum_{j=1}^m \sum_{i=1}^3 |\alpha_{ij}|^2 \right) |s_4|^2 \quad (12.46)$$

12.4.2 Simulation of BER Performance of STBCs Using MATLAB

The following MATLAB program 12.2 and MATLAB function programs 12.3, 12.4, 12.5, and 12.6 are used to simulate the BER performance of QPSK and 16-QAM for STBC (3×4), Alamouti (2×2), and Alamouti (2×1)

Program 12.2: “STBC simulation.m” for BER performance comparison of STBC (3×4), Alamouti (2×2), and Alamouti (2×1) in Rayleigh fading channel

```
%STBCsimulation.m
% BER performance of Alamouti(2x1),Alamouti(2x2),STBC(3x4)
clear; clc;
framesize=128; Packets=4*1024; % Number of frames/packet and Number of
packets
b=input('enter the value b=1 for BPSK,b=2 for QPSK,b=4 for 16-QAM');
M=2^b;
Eb_N0_dB=[0:2:20];
for i=1:length(Eb_N0_dB)
    EbN0=Eb_N0_dB(i); sigma=sqrt(0.5/(10^(EbN0/10)));
    for i1=1:Packets
        [Xo1 Xe1]=stbc2by1(framesize,Packets,2,M,sigma,b);
        [Xo2 Xe2]=stbc2by2(framesize,Packets,2,M,sigma,b);
        [Xo3 Xe3]=stbc3by4(framesize,Packets,3,M,sigma,b);
        packet_error1(i1) = sum(sum(Xo1~=Xe1));
        packet_error2(i1) = sum(sum(Xo2~=Xe2));
        packet_error3(i1) = sum(sum(Xo3~=Xe3));
    end % End of FOR loop for packets
    BER1(i) = sum(packet_error1)/(Packets*framesize*b);
    BER2(i) = sum(packet_error2)/(Packets*framesize*b);
    BER3(i) = sum(packet_error3)/(Packets*framesize*b);
end % End of FOR loop for Eb_N0_dB
semilogy(Eb_N0_dB,BER1,'x'),hold on;
semilogy(Eb_N0_dB,BER2,'+'), hold on;
semilogy(Eb_N0_dB,BER3,'o'),axis([Eb_N0_dB([1 end]) 1e-6 1e0]);
grid on; xlabel('Eb/N0(dB)'); ylabel('BER');
legend('Alamouti(2x1)', 'Alamouti(2x2)', 'STBC(3x4)')
```

Program 12.3: MATLAB function for stbc2by1

```

function [So Se]=stbc2by1(framesize,Packets,NT,M,sigma,b)
inp_bits=randint(framesize*b,M);
bitsequence=inp_bits.';
tmp=[]; tmp1=[];
for i=1:NT
[tmp1,symbols] = modulation(bitsequence(i,:),b); tmp=[tmp; tmp1];
end
S=tmp.';
S1=S; S2=[-conj(S(:,2)) conj(S(:,1))];
H=(randn(framesize,NT)+j*randn(framesize,NT))/sqrt(2);
Habs=sum(abs(H).^2,2);
y11 = sum(H.*S1,2)/sqrt(2) + sigma*(randn(framesize,1)+j*randn(framesize,1));
y12 = sum(H.*S2,2)/sqrt(2) + sigma*(randn(framesize,1)+j*randn(framesize,1));
S1e = y11 .*conj(H(:,1)) + conj(y12).*H(:,2);
S2e = y11.*conj(H(:,2)) - conj(y12).*H(:,1);
for m=1:M
eta = (-1+sum(Habs,2))*abs(symbols(m))^2;
d1(:,m) = abs(sum(S1e,2)-symbols(m)).^2 + eta;
d2(:,m) = abs(sum(S2e,2)-symbols(m)).^2 + eta;
end
[y1,i1]=min(d1,[],2); S1d=symbols(i1).'; clear d1
[y2,i2]=min(d2,[],2); S2d=symbols(i2).'; clear d2
Sd =[S1d S2d]; So=S>0; Se=Sd>0;

```

Program 12.4: MATLAB function for stbc2by2

```

function [So Se]=stbc2by2(framesize,Packets,NT,M,sigma,b)
inp_bits=randint(framesize*b,M);
bitsequence=inp_bits.';
tmp=[]; tmp1=[];
for i=1:NT
[tmp1,symbols] = modulation(bitsequence(i,:),b); tmp=[tmp; tmp1];
end
S=tmp.';
S1=S; S2=[-conj(S(:,2)) conj(S(:,1))];
H=(randn(framesize,2*NT)+j*randn(framesize,2*NT))/sqrt(2);Habs=sum(abs(H).^2,2);
y11 = sum(H(:,1:2).*S1,2)/sqrt(2) + sigma*(randn(framesize,1)+j*randn(framesize,1));
y12 = sum(H(:,1:2).*S2,2)/sqrt(2) + sigma*(randn(framesize,1)+j*randn(framesize,1));
y21 = sum(H(:,3:4).*S1,2)/sqrt(2) + sigma*(randn(framesize,1)+j*randn(framesize,1));
y22 = sum(H(:,3:4).*S2,2)/sqrt(2) + sigma*(randn(framesize,1)+j*randn(framesize,1));
S1e = y11 .*conj(H(:,1)) + conj(y12).*H(:,2)+y21.*conj(H(:,3)) + conj(y22).*H(:,4);
S2e = y11.*conj(H(:,2)) - conj(y12).*H(:,1)+y21.*conj(H(:,4)) - conj(y22).*H(:,3);
for m=1:M
eta = (-1+sum(Habs,2))*abs(symbols(m))^2;
d1(:,m) = abs(sum(S1e,2)-symbols(m)).^2 + eta;
d2(:,m) = abs(sum(S2e,2)-symbols(m)).^2 + eta;
end
[y1,i1]=min(d1,[],2); S1d=symbols(i1).'; clear d1
[y2,i2]=min(d2,[],2); S2d=symbols(i2).'; clear d2
Sd =[S1d S2d]; So=S>0; Se=Sd>0;

```

Program 12.5: MATLAB function for stbc3by4

```

function [So Se]=stbc3by4(framesize,packets,NT,M,sigma,b)
inp_bits=randint(framesize*b,M);
bitsequence=inp_bits.';

temp=[];
temp1=[];
for i=1:4
    [temp1,symbols]=modulation(bitsequence(i,:),b);
    temp=[temp; temp1];
end
S=temp.';

% Block signals in the l-th time slot % Block coding for G3 STBC
S1=S(:,1:3); S5=conj(S1);
S2 = [-S(:,2) S(:,1) -S(:,4)]; S6=conj(S2);
S3 = [-S(:,3) S(:,4) S(:,1)]; S7=conj(S3);
S4 = [-S(:,4) -S(:,3) S(:,2)]; S8=conj(S4);
for n=1:NT
Hr(n,:,:)=(randn(framesize,NT)+j*randn(framesize,NT))/sqrt(2);
end
for n=1:NT
H = reshape(Hr(n,:,:),framesize,NT);
Hc=conj(H);
Habs(:,n)= sum(abs(H).^2,2);
y1 = sum(H.*S1,2)/sqrt(NT)+sigma*(randn(framesize,1)+j*randn(framesize,1));
y2 = sum(H.*S2,2)/sqrt(NT)+sigma*(randn(framesize,1)+j*randn(framesize,1));
y3 = sum(H.*S3,2)/sqrt(NT)+sigma*(randn(framesize,1)-j*randn(framesize,1));
y4 = sum(H.*S4,2)/sqrt(NT)+sigma*(randn(framesize,1)-j*randn(framesize,1));
y5 = sum(H.*S5,2)/sqrt(NT)+sigma*(randn(framesize,1)+j*randn(framesize,1));
y6 = sum(H.*S6,2)/sqrt(NT)+sigma*(randn(framesize,1)+j*randn(framesize,1));
y7 = sum(H.*S7,2)/sqrt(NT)+sigma*(randn(framesize,1)-j*randn(framesize,1));
y8 = sum(H.*S8,2)/sqrt(NT)+sigma*(randn(framesize,1)+j*randn(framesize,1));
Se(:,n,1)=y1.*Hc(:,1)+y2.*Hc(:,2)+y3.*Hc(:,3)+ conj(y5).*H(:,1) +
conj(y6).*H(:,2)+ conj(y7).*H(:,3);
Se(:,n,2)=y1.*Hc(:,2)-y2.*Hc(:,1)+y4.*Hc(:,3)+ conj(y5).*H(:,2) -
conj(y6).*H(:,1)+ conj(y8).*H(:,3);
Se(:,n,3)=y1.*Hc(:,3)-y3.*Hc(:,1)-y4.*Hc(:,2)+ conj(y5).*H(:,3) -
conj(y7).*H(:,1)- conj(y8).*H(:,2);
Se(:,n,4)=-y2.*Hc(:,3)+y3.*Hc(:,2)-y4.*Hc(:,1)+(-conj(y6)).*H(:,3) +
conj(y7).*H(:,2)- conj(y8).*H(:,1);
end
for m=1:M
tmp = (-1+sum(Habs,2))*abs(symbols(m))^2;
for i=1:4
d(:,m,i)= abs(sum(Se(:, :, i),2)-symbols(m)).^2 + tmp;
end
end
Sd = [];
for n=1:4, [yn,in]=min(d(:, :, n),[],2); Sd = [Sd symbols(in).'];
end
So=S>0; Se=Sd>0;

```

Program 12.6: MATLAB function for BPSK, QPSK, and 16-QAM mapping

```

function [modsymbols,symbols] = modulation(inp,b)
bitlength = length(inp);
switch b
case {1} % BPSK modulation
symbols=exp(j*[0 -pi]); symbols=symbols([1 0]+1);
modsymbols=symbols(inp+1);
case {2} % QPSK modulation
symbols = exp(j*pi/4*[-3 3 1 -1]); symbols=symbols([0 1 3 2]+1);
inp=reshape(inp,b,bitlength/b);
modsymbols=symbols([2 1]*inp+1);
case {4} % 16-QAM modulation
m=0;
for k=-3:2:3
for l=-3:2:3
m=m+1; symbols(m) = (k+j*l)/sqrt(10); % power normalization
end
end
symbols = symbols([0 1 3 2 4 5 7 6 12 13 15 14 8 9 11 10]+1);
inp = reshape(inp,b,bitlength/b);
modsymbols = symbols([8 4 2 1]*inp+1);
% maps transmitted bits into 16QAM symbols
otherwise
disp ('Unimplemented modulation!');
end

```

The BER performance obtained by using above programs for QPSK and 16-QAM with STBC (3×4), Alamouti (2×2), and Alamouti (2×1) is shown in Figs. 12.6 and 12.7, respectively.

From Figs. 12.6 and 12.7, it can be seen that the BER performance of STBC (3×4) is better than Alamouti (2×2) and Alamouti (2×1).

12.5 Space-Time Trellis Coding

In contrast to STBCs, STTCs provide both coding gain and diversity gain and have a better bit error rate performance. However, STTCs are more complex than STBCs to encode and decode.

In [2], Tarokh et al. derived the design criteria for STTCs over slow frequency non-selective fading channels. The design criteria were shown to be determined by the distance matrices constructed from pairs of distinct code words. The minimum rank of the distance matrices was used to determine the diversity gain, and the minimum distance of the distance matrices was used to determine the coding gain [3]. The system model for STTC modulation is shown in Fig. 12.8.

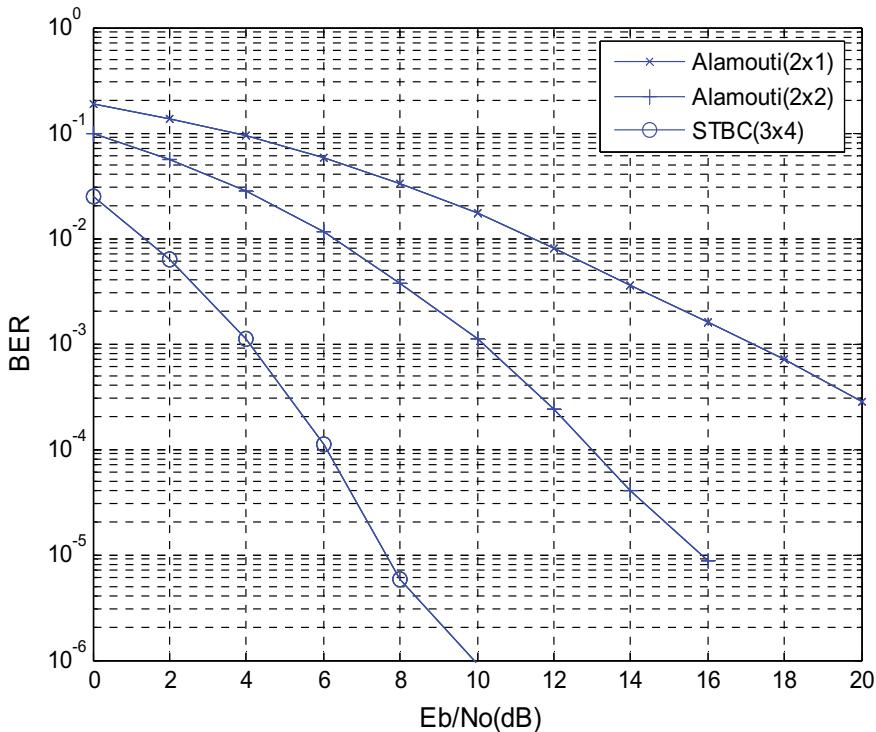


Fig. 12.6 BER performance for QPSK with STBC (3 × 4), Alamouti (2 × 2), and Alamouti (2 × 1)

12.5.1 Space-Time Trellis Encoder

Let

$$I_t = \begin{bmatrix} I_t^1 \\ I_t^2 \\ \vdots \\ I_t^m \end{bmatrix}$$

denote input data symbol of $m = \log_2 M$ bits, which is input to the encoder at time $t = 0, 1, 2, \dots$, then, a sequence of input data symbols is represented as

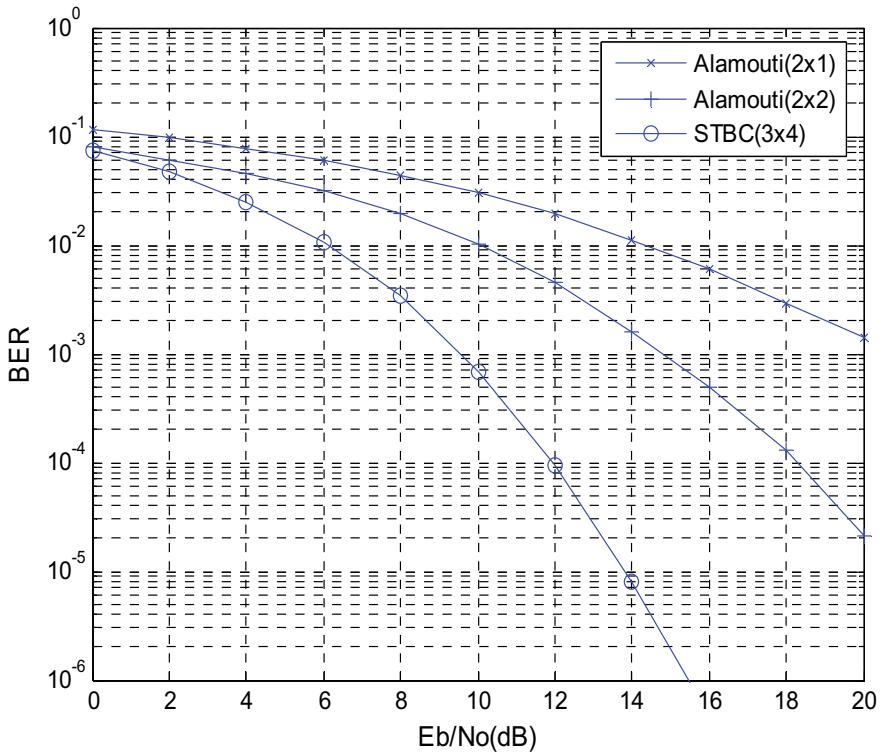


Fig. 12.7 BER performance for 16-QAM with STBC (3×4), Alamouti (2×2), and Alamouti (2×1)

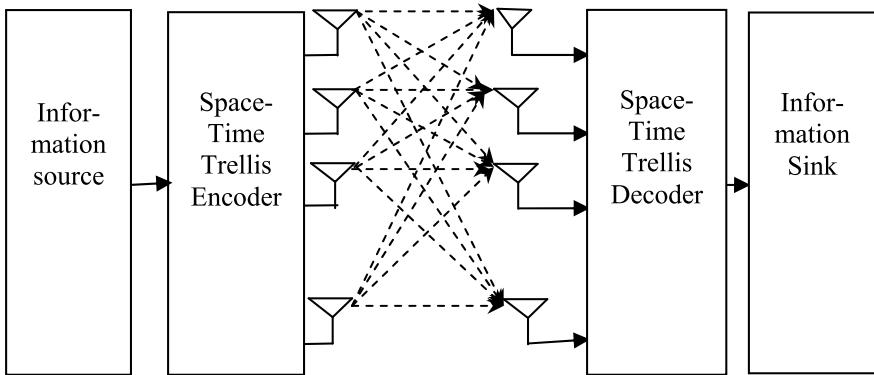


Fig. 12.8 Space-time trellis code system model

$$I = \begin{bmatrix} I_0^1 & I_1^1 & \cdots & I_t^1 & \cdots \\ I_0^2 & I_1^2 & \cdots & I_t^2 & \cdots \\ \vdots & \vdots & \ddots & \vdots & \\ I_0^m & I_1^m & \cdots & I_t^m & \cdots \end{bmatrix} \quad (12.47)$$

The STTC encoder can be considered as a convolutional encoder with the memory size of v_k delay units for the k th branch for each output symbol. Let $\{v_k\}_{k=1}^m$ denote the size of memory used to store the k th branch metrics that is calculated as

$$v_k = \frac{v + k - 1}{\log_2 M} \quad (12.48)$$

where $\lfloor x \rfloor$ denotes the largest integer smaller than x . v is the size of total required memory for the space-time trellis code, that is,

$$v = \sum_{k=1}^m v_k \quad (12.49)$$

Then, the output of the STTC encoder is specified by the following generator polynomials:

$$\begin{aligned} a^1 &= [(a_{0,1}^1, a_{0,2}^1, \dots, a_{0,N_r}^1), (a_{1,1}^1, a_{1,2}^1, \dots, a_{1,N_r}^1), \dots, (a_{v_1,1}^1, a_{v_1,2}^1, \dots, a_{v_1,N_r}^1)] \\ a^2 &= [(a_{0,1}^2, a_{0,2}^2, \dots, a_{0,N_r}^2), (a_{1,1}^2, a_{1,2}^2, \dots, a_{1,N_r}^2), \dots, (a_{v_2,1}^2, a_{v_2,2}^2, \dots, a_{v_2,N_r}^2)] \\ &\vdots \\ a^m &= [(a_{0,1}^m, a_{0,2}^m, \dots, a_{0,N_r}^m), (a_{1,1}^m, a_{1,2}^m, \dots, a_{1,N_r}^m), \dots, (a_{v_m,1}^m, a_{v_m,2}^m, \dots, a_{v_m,N_r}^m)] \end{aligned} \quad (12.50)$$

where $a_{j,i}^k$ denotes M-PSK symbols, $k = 1, 2, \dots, m$; $j = 1, 2, \dots, v_k$; $i = 1, 2, \dots, N_T$. Let y_t^i denote the outputs of the STTC encoder for the i th transmit antenna at time t , $i = 1, 2, \dots, N_T$, which are given as

$$X_t^i = \sum_{k=1}^m \sum_{j=0}^{v_k} a_{j,i}^k I_{tj}^k \bmod M \quad (12.51)$$

Space-time trellis-encoded M-PSK symbols are now expressed as

$$X = [X_0 X_1 \dots X_t \dots] = \begin{bmatrix} x_0^1 & x_1^1 & \dots & x_t^1 & \dots \\ x_0^2 & x_1^2 & \dots & x_t^2 & \dots \\ \vdots & \vdots & \ddots & \vdots \\ x_0^{N_T} & x_1^{N_T} & \dots & x_t^{N_T} & \dots \end{bmatrix} \quad (12.52)$$

where $X_t = [x_t^1 x_t^2 \dots x_t^{N_T}]^T$ is the output of the encoder that is composed of N_T M-PSK symbols, $t = 0, 1, 2, \dots$. Figure 12.9 shows an example of the STTC encoder for $N_T = 2$, $m = 3$, and $v = 3$.

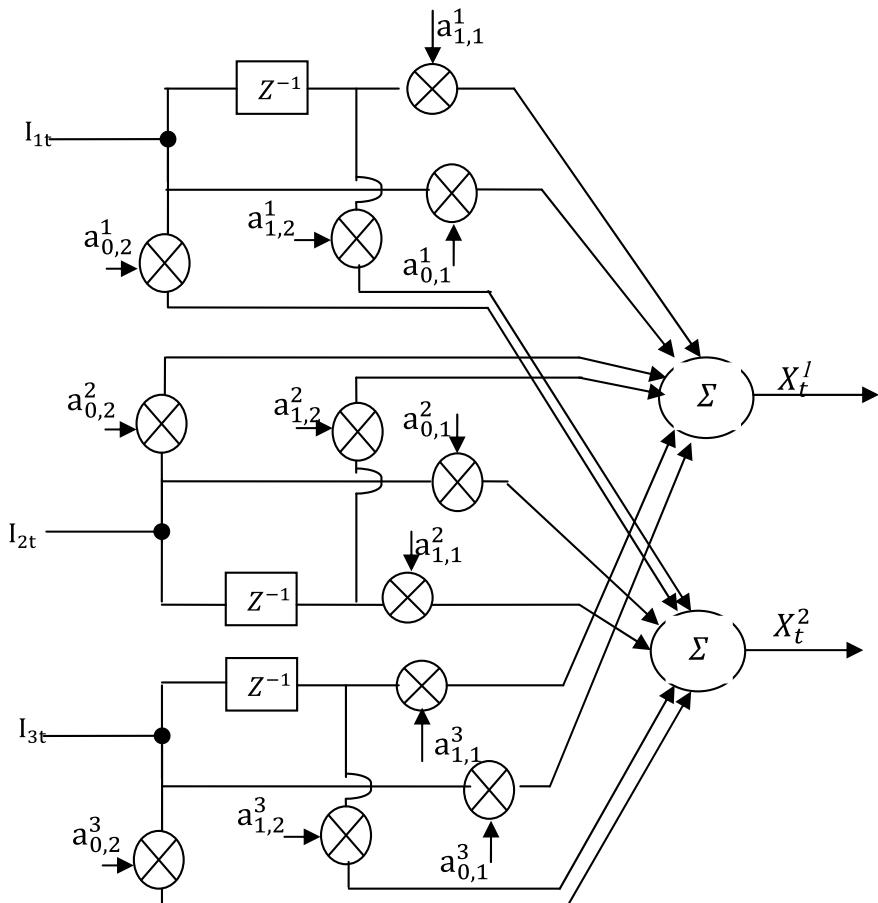


Fig. 12.9 8-state, 8-PSK encoder structure

Table 12.1 Coefficient pairs for 4-PSK, 4-, 8-, and 16-state STTC

V	$(a_{0,1}^1, a_{0,2}^1)$	$(a_{1,1}^1, a_{1,2}^1)$	$(a_{2,1}^1, a_{2,2}^1)$	$(a_{0,1}^2, a_{0,2}^2)$	$(a_{1,1}^2, a_{1,2}^2)$	$(a_{2,1}^2, a_{2,2}^2)$	$\det(v)$	$\text{Tr}(v)$
2	(0,2)	(2,0)	–	(0,1)	(1,0)	–	4	4
3	(0,2)	(2,0)	–	(0,1)	(1,0)	(2,2)	12	8
4	(0,2)	(2,0)	(0,2)	(0,1)	(1,2)	(2,0)	12	8

Table 12.2 Coefficient pairs for 8-PSK, 8-state STTC

V	$(a_{0,1}^1, a_{0,2}^1)$	$(a_{1,1}^1, a_{1,2}^1)$	$(a_{2,1}^1, a_{2,2}^1)$	$(a_{0,1}^2, a_{0,2}^2)$	$(a_{1,1}^2, a_{1,2}^2)$	$(a_{0,1}^3, a_{0,2}^3)$	$(a_{1,1}^3, a_{1,2}^3)$	$\det(v)$	$\text{Tr}(v)$
3	(0,4)	(4,0)	(0,2)	(2,0)	(0,1)	(5,0)	2	4	

Some of the coefficients for 4-PSK STTC and 8-PSK STTC codes [8] are summarized in Tables 12.1 and 12.2, respectively.

The Viterbi algorithm can be used for decoding the space-time trellis-coded systems. In the Viterbi algorithm, the branch metric is given by the following squared Euclidian distance:

$$\sum_{t=1}^T \sum_{j=1}^{N_R} \left| y_t^j - \sum_{i=1}^{N_T} h_{j,i} x_t^i \right|^2 \quad (12.53)$$

where y_t^j is the received signal at the j th receive antenna during t th symbol period, and $h_{j,i}$ is the channel gain between the i th transmit antenna and j th receive antenna. Using the branch metric in Eq. (12.53), a path with the minimum accumulated Euclidean distance is selected for the detected sequence of transmitted symbols.

12.5.1.1 4-State QPSK Space-Time Trellis Encoder

STTCs can be represented and analyzed in their trellis form or by their generator matrix, G . For example, consider the 4-PSK signal constellation shown in Fig. 12.10a, where the signal points are labeled as 0, 1, 2, and 3.

The 4-state trellis structure is shown in Fig. 12.10b for a rate of 2 b/s/Hz.

The input signal can take on any value from the signal constellation (in this case 0, 1, 2, or 3); they are shown on the trellis diagram on the transition branches. In general, for each state, the first transition branch to state 0 results from input 0, the second transition branch to state 1 results from input 1, and so on. The output depends on the input and on the current state. The states are labeled on the right. The labels on the left of the trellis represent the possible outputs from that state. The leftmost output is assumed to be the output for the first trellis branch for that particular state,

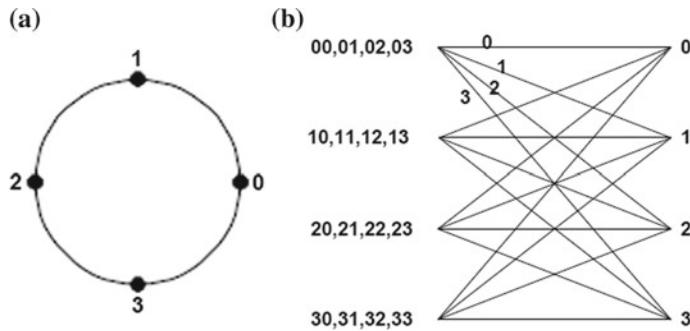


Fig. 12.10 a 4-PSK signal constellation. b 4-state, 4-PSK trellis diagram

and the second leftmost label is assumed to be the output for the second trellis branch for the same state, and so on. These assumptions were verified to be correct and can be manually traced through the encoder structure.

It was proved in [2] that the above code provides a diversity gain of 2 (assuming one receive antenna) and has a minimum determinant of 2 [4].

The encoder structure for the 4-state ($v = 2$) trellis, QPSK scheme with two transmit antennas is shown in Fig. 12.11.

At time t , two binary inputs I_t^1 and I_t^2 are fed into the branches of the encoder with I_t^1 being the MSB. The memory order of the upper and lower branches are V_1

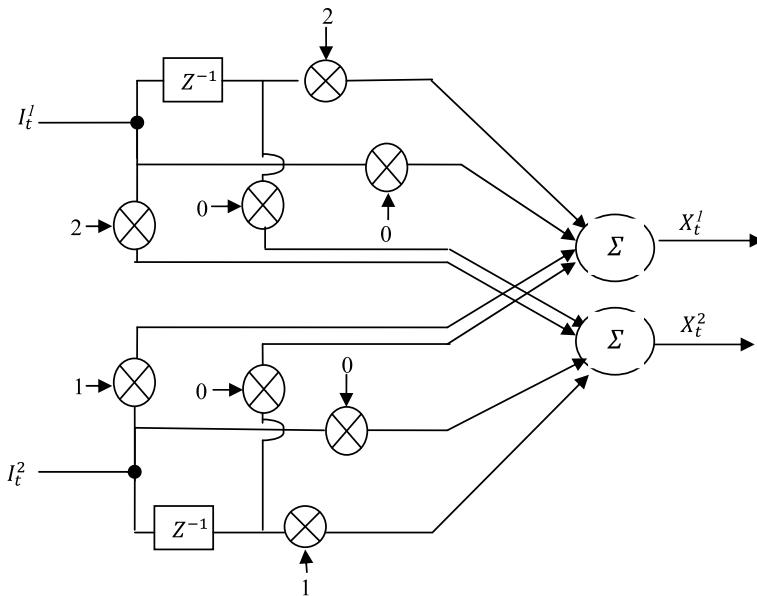


Fig. 12.11 4-state, 4-PSK encoder structure

and V_2 , respectively, where $V = V_1 + V_2$, and hence, the number of states is 2^V . V_i is calculated as

$$V_i = \left\lfloor \frac{V+i-1}{2} \right\rfloor, i = 1, 2 \quad (12.54)$$

where $\lfloor X \rfloor$ denotes the largest integer smaller than or equal to X . For each branch, the output is the sum of the current input scaled by a coefficient and the previous input scaled by another coefficient. Each of the different coefficients in the coefficient pairs, (0,2), (2,0), (0,1), and (1,0), applied to I_t^1 and I_t^2 , respectively.

By using Eq. (12.52), we can get the output values as follows

$$X_t^1 = (2I_{t-1}^1 + I_{t-1}^2) \bmod 4 \quad (12.55)$$

$$X_t^2 = (2I_t^1 + I_t^2) \bmod 4 \quad (12.56)$$

X_t^1 and X_t^2 are transmitted simultaneously on the first and second antenna, respectively. From Eqs. (12.55) and (12.56), it can be seen that $X_t^1 = X_{t-1}^1$, that is, the signal transmitted from the first antenna is a delayed version of the transmitted signal from the second transmit antenna. Note that the output X_t^2 at time t becomes the encoder state at time $(t+1)$ in this particular example.

Example 12.3 Consider the STTC encoder shown in Fig. 12.11 determine the trellis-encoded symbol stream if the two input bit sequences are

$$\begin{bmatrix} I_t^1 \\ I_t^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Figure 12.11 shows a structure of the STTC encoder for this example. The encoder state at time t is $(I_{t-1}^1 I_{t-1}^2)$ or $2I_{t-1}^1 + I_{t-1}^2$. The output for the i th transmit antenna at time t is calculated as

$$X_t^1 = (2I_{t-1}^1 + I_{t-1}^2) \bmod 4$$

and

$$X_t^2 = (2I_t^1 + I_t^2) \bmod 4$$

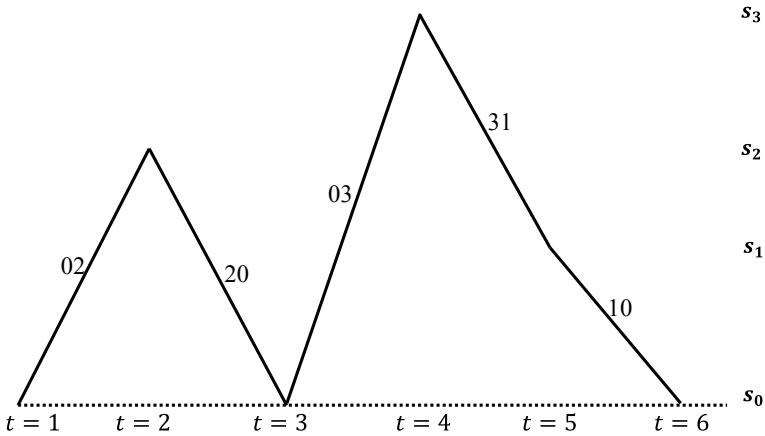


Fig. 12.12 4-state, 4-PSK encoder's output for Example 11.1

$$\mathbf{Y} = \begin{bmatrix} X_t^1 \\ X_t^2 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 0 & 3 & 1 \\ 2 & 0 & 3 & 1 & 0 \end{bmatrix}$$

Figure 12.12 shows the corresponding trellis diagram, in which the branch labels indicate two output symbols, X_t^1 and X_t^2 .

At time $t = 1$, we have $x_t^1 = 0$ and $x_t^2 = 2$. Therefore, 1 and -1 are transmitted from first and second antennas, respectively.

At time $t = 2$, we have $x_t^1 = 2$ and $x_t^2 = 0$. Therefore, -1 and 1 are transmitted from first and second antennas, respectively.

At time $t = 3$, we have $x_t^1 = 0$ and $x_t^2 = 3$. Therefore, 1 and $-j$ are transmitted from first and second antennas, respectively.

At time $t = 4$, we have $x_t^1 = 3$ and $x_t^2 = 1$. Therefore, $-j$ and j are transmitted from first and second antennas, respectively.

At time $t = 5$, we have $x_t^1 = 1$ and $x_t^2 = 0$. Therefore, j and 1 are transmitted from first and second antennas, respectively.

12.5.1.2 8-State 8-PSK Space-Time Trellis Encoder

The 8-state 8-PSK signal constellation and trellis diagram are shown in Figs. 12.13 and 12.14, for a rate of 3 b/s/Hz.

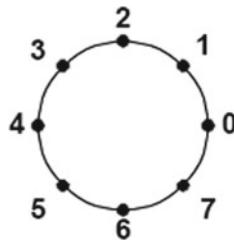


Fig. 12.13 8-PSK signal constellation

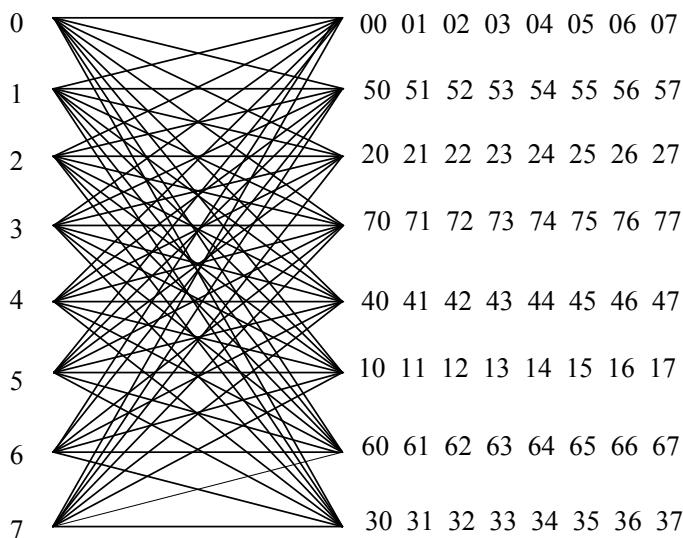


Fig. 12.14 8-state 8-PSK trellis diagram

12.5.2 *Simulation of BER Performance of 4-state QPSK STTC Using MATLAB*

The following MATLAB program 12.7 and MATLAB function programs 12.8 through 12.15 are used to simulate the BER performance of 4-state QPSK STTC.

Program 12.7: Program for Space-Time Trellis Code (STTC) for 4-state QPSK

```
clear all,clc,
Eb_N0_dB=0:2:20; iter= 1000;
g1=[0 2;2 0]; % 1-st generator
g2=[0 1;1 0]; % 2-nd generator
M = 4; % M-PSK = QPSK
BER1=qpsksttc(1,g1,g2,M,Eb_N0_dB,iter );
BER2=qpsksttc(2,g1,g2,M,Eb_N0_dB,iter);
BER4=qpsksttc(4,g1,g2,M,Eb_N0_dB,iter);
figure; % BER vs Eb/No in logarithmic value
semilogy(Eb_N0_dB,BER1,'-');hold on;
semilogy(Eb_N0_dB,BER2,'-+');hold on;
semilogy(Eb_N0_dB,BER4,'-*');
grid on; xlabel('Eb/No(dB)'); ylabel('BER');
legend('Nrx =1','Nrx =2','Nrx =4');
```

Program 12.8: MATLAB function for qpsksttc

```
function BER=qpsksttc(Nrx,g1,g2,M,Eb_N0_dB,iter)
Ntx=2;
data=randint(1,48,2);
% ===== Encoder STTC =====
[s,moddata,v] = sttcenc(g1,g2,data,M);
modinp = moddata;
for i1=1:iter
    H=[randn(Nrx,Ntx) + i*randn(Nrx,Ntx)]; % MIMO channel
    txs=H*modinp;
    for k = 1:length(Eb_N0_dB)
        % ===== Noise Addition =====
        rxs=awgn(txs,Eb_N0_dB(k),'measured');
        % ===== STTC Detection =====
        symbol = symbolmap(g1,g2,M);
        sum1 = MaxLike(rxs,symbol,H,Nrx);
        [data_detect] = viterbi (sum1,v,M,g1,g2);
        % ===== BER computation =====
        error(1,k) = sum(data~=data_detect); % BER
    end
    total_error(i1,:)=error;
end
BER = sum(total_error)/(length(data)*iter);
```

Program 12.9: MATLAB function for sttcenc

```

function [s,moddata,v] = sttcenc(g1,g2,data,M)
[a b]= size(data);
v1 = length(g1)-1;
v2 = length(g2)-1;
v = v1+v2; % memory degree
m =log2 (M);
% serial to paralel
ncol = b/m;
for index=0:ncol -1
    c(:,index+1) = data(1,(m * index)+1:m*(index+1));
end
temp1 = zeros (m,v);
for k=0:ncol -1 % time
    % initialization
    for in=0:v-2
        temp1(:,v-in)=temp1(:,v-in-1);
    end
    temp1(:,1)=c(:,k+1);
    % symbol = g*c
    for l=1:m
        for j=0:v1
            temp2(l,j+1)= g1(l,j+1)*temp1(1,j+1);
        end
        for j=0:v2
            temp3(l,j+1)= g2(l,j+1)*temp1(2,j+1);
        end
        shat(l,k+1) = sum(sum(temp2)) + sum(sum(temp3));
        temp2=[];temp3=[];
    end
    temp1;
end
s = mod(shat,M);
[n m]=size(s);
moddata=[];
% Mapper QPSK from symbol STTC
for a=1:n
    for b=1:m
        if s(a,b)==[0]
            moddata(a,b) = 1+i;
        elseif s(a,b)==[1]
            moddata(a,b) = 1-i;
        elseif s(a,b)==[2]
            moddata(a,b) = -1-i;
        elseif s(a,b)==[3]
            moddata(a,b) = -1+i;
        end;
    end;
end;

```

Program 12.10: MATLAB function for symbolmap

```

function symbol = symbolmap(g1,g2,M)
[trellis1,trellis2,statego]=gen2trellis(g1,g2,M);
[c d] = size (trellis1);
for l=0:c-1
    for j=1:d
        map1(l*M+j,1) = trellis1 (l+1,j);
        map1(l*M+j,2) = trellis2 (l+1,j);
    end
end
[n m]=size(map1);
map1;
symbol = [];
for c=1:n
    for d=1:m
        if map1(c,d)== 0
            symbol(c,d) = 1+i;
        elseif map1(c,d)== 1
            symbol(c,d) = 1-i;
        elseif map1(c,d)== 2
            symbol(c,d) = -1-i;
        elseif map1(c,d)== 3
            symbol(c,d) = -1+i;
        end;
    end;
end;

```

Program 12.11: MATLAB function for MaxLike

```

function dist = MaxLike(symrec,symref,H,Nrx);
for c=1:length(symrec)
    for p=1:size(symref,1)
        for d=0:Nrx-1
            disthat(1,d+1)=norm((H(d+1,:)*symref(p,:).' -
symrec(d+1,c)),'fro');
        end
        dist(p,c) = sum(disthat)/Nrx;
    end
end

```

Program 12.12: MATLAB function for viterbi

```

function [data] = viterbi (distance,v,M,g1,g2)
nstates = 2^v;
dist = [];
[brs kol]=size (distance);
[trellis1,trellis2,statego]=gen2trellis(g1,g2,M);
for a = 1:kol
    if a == 1
        dist(:,a) = distance(1:nstates,a);
    else
        disthat = [];
        for b=0:M-1
            for bb=1:nstates
                if bb < 5
                    disthat(bb,b+1) = dist(b*(v-1)+1,a-1)+ distance(b*nstates+bb,a);
                else
                    disthat(bb,b+1) = dist(b*(v-1)+2,a-1)+ distance(b*nstates+bb,a);
                end
            end
            disthat;
            for b=1:nstates
                dist(b,a)=min (distrath(b,:));
            end
        end
        end
        state= 0;
        for a = 1:kol
            p = statego(state+1,1)+1;
            q = statego(state+1,4)+1;
            [mindist(:,a) ind(:,a)] = min (dist(p:q,a));
            if p ~= 1
                ind(:,a)=ind(:,a)+4;
            end
            state = ind(:,a)-1;
        end
        survpath = ind -1;
        demap = [0 0;0 1;1 0;1 1]';
        for a = 1:kol
            if a == 1
                srx(1,a) = trellis1(1,ind(1,a));
                srx(2,a) = trellis2(1,ind(1,a));
                data(1,2*(a-1)+1:2*(a-1)+2)= demap(:,ind(1,a))';
            else
                dd =find (survpath(1,a) == statego(ind(1,a-1),:));
                srx(1,a) = trellis1(ind(1,a-1),dd);
                srx(2,a) = trellis2(ind(1,a-1),dd);
                data(1,2*(a-1)+1:2*(a-1)+2)= demap(:,dd)';
            end
        end
    end
end

```

Program 12.13: MATLAB function for gen2trellis

```

function [trellis1,trellis2,statego]=gen2trellis(g1,g2,M)
% function to change generator code to trellis
% trellis1 = trellis antenna 1
% trellis2 = trellis antenna 2
% statoeg = state moving
v1 = length(g1)-1;
v2 = length(g2)-1;
v = v1+v2;
nstates = 2^v;
m = log2 (M);
state = 0:nstates-1;
for a=1:nstates
    for b=1:4
        input = [getbits(state (1,a),nstates/2),getbits(state(1,b),2)];
        % serial to paralel
        ncol = length (input)/2;
        for index=0:ncol-1
            c(:,index+1) = input(1,(m * index)+1:m*(index+1));
        end
        temp1 = zeros (m,v);
        for k=0:ncol-1 % time
            % initialization
            for in=0:v-2
                temp1(:,v-in)=temp1(:,v-in-1);
            end
            temp1(:,1)=c(:,k+1);
            % symbol = g*c
            for l=1:m
                for j=0:v1
                    temp2(l,j+1)= g1(l,j+1)*temp1(1,j+1);
                end
                for j=0:v2
                    temp3(l,j+1)= g2(l,j+1)*temp1(2,j+1);
                end
                shat(l,k+1) = sum(sum(temp2)) + sum(sum(temp3));
                temp2=[];temp3=[];
            end
            temp1;
        end
        s = mod(shat,M);
        trellis1(a,b) = s(1,ncol);
        trellis2(a,b) = s(2,ncol);
        statoeg(a,b) = bit2num(input(1,(length(input)-v2):length(input)));
    end
end

```

Program 12.14: MATLAB function for getbits

```
function [bits] = getbits(x, n)
bits = zeros(1, n);
ind = 1;
while (x~=0)
    bits(ind) = mod(x,2);
    x = floor(x/2);
    ind = ind + 1;
end
bits = fliplr(bits);
```

Program 12.15: MATLAB function for bit2num

```
function [y] = bit2num(x)
y = 0; mul = 1;
for i=(length(x):-1:1)
    y = y + mul*x(i);
    mul = mul*2;
end
```

The BER performance obtained by using above programs for 4-state QPSK STTC is shown in Fig. 12.15.

From Fig. 12.15, it is observed that the STTC with four receiving antennas out performs the STTC with one and two receiving antennas.

12.6 MIMO-OFDM Implementation

A MIMO-OFDM system is shown in Fig. 12.16 where OFDM utilizes N_T transmit antennas, N_R receive antennas, and N_c subcarriers per antenna. MIMO-OFDM can be implemented as space-time-coded OFDM (ST-OFDM), space-frequency coded OFDM (SF-OFDM), and space-time-frequency coded OFDM (STF-OFDM). Let x_n^μ (i) be the data symbol transmitted on the i th subcarrier (frequency bin) from the μ th transmit antenna during the n th OFDM symbol interval. Then, the difference among these coded systems lies in how x_n^μ (i) are generated from the information symbols S_n [9].

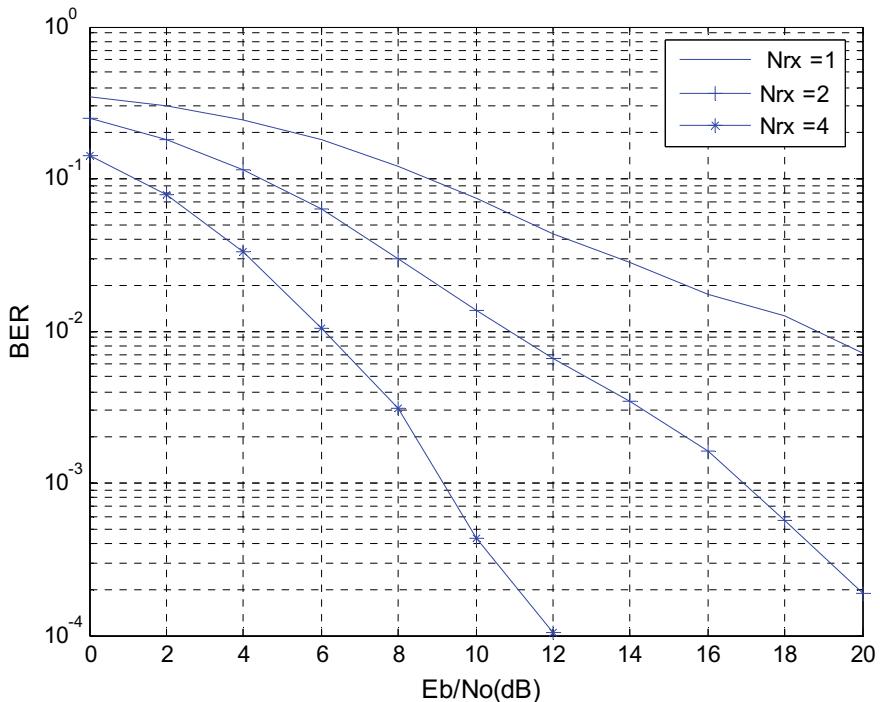


Fig. 12.15 BER performance of 4-state QPSK STTC

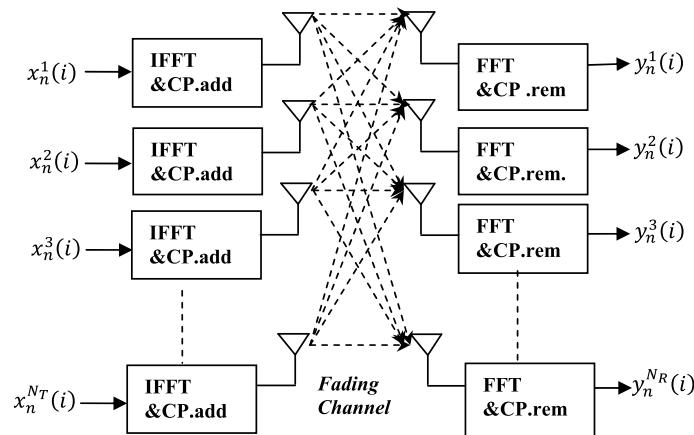


Fig. 12.16 MIMO OFDM system

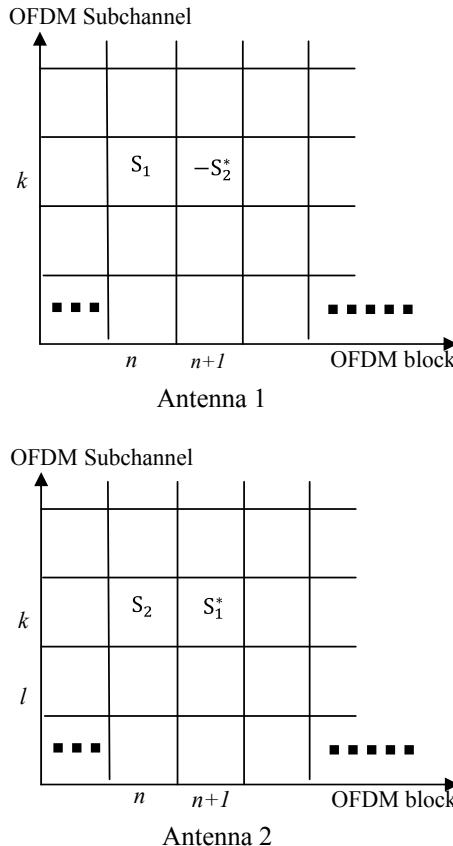


Fig. 12.17 ST coding

12.6.1 Space-Time-Coded OFDM

The ST coding for a MIMO-OFDM with two transmit antennas is illustrated in Fig. 12.17. Two information symbols s_1 and $-s_2^*$ are sent through sub channel k of antenna 1 in OFDM blocks n and $n + 1$, respectively. Meanwhile, s_2 and s_1^* are sent through subchannel k of antenna 2 in OFDM blocks n and $n + 1$, respectively.

12.6.2 Space-Frequency Coded OFDM

In space-time-coded OFDM, the frequency diversity and the correlation among different subcarriers are ignored. The strategy, which consists of coding across antennas and different subcarriers of OFDM, is called SF coded OFDM [10]. The schematic

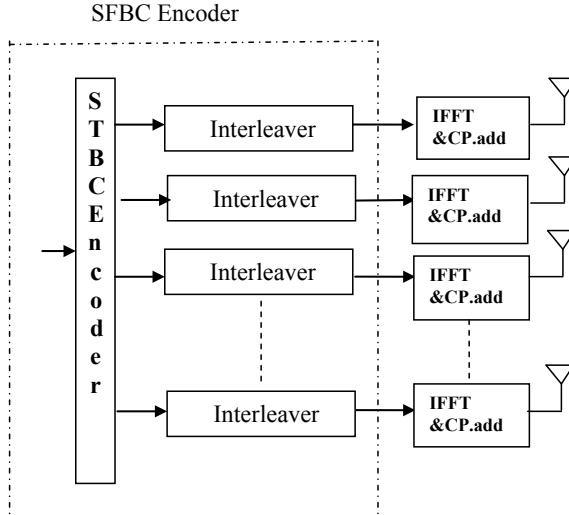


Fig. 12.18 Block diagram of SFBC coded OFDM

diagram of SF coded OFDM is shown in Fig. 12.18. The STBC encoder generates $N_c \times N_T$ symbols for each OFDM block (time slot). One data burst therefore consists of N_c vectors of size $N_T \times 1$ or equivalently one spatial OFDM symbol. The channel is assumed to be constant over at least one OFDM symbol. The interleaver transmits the (l, n) symbol on the l th subcarrier of the n th antenna [11].

The SF coding for two transmit antennas can be realized in a straight forward way by spreading directly the Alamouti code over two subchannels in one OFDM block. An example of SF coding for two transmit antennas is shown in Fig. 12.19. The two symbols S_1 and $-S_2^*$ are sent from subchannels k and l of the same OFDM block n at antenna 1, respectively, where k and l denote the indices of two separated subchannels. Meanwhile, S_2 and S_1^* are sent from subchannels k and l of the same OFDM block n at antenna 2, respectively [12].

12.6.3 Space-Time-Frequency Coded OFDM

In STF coding, each x_n^μ (i) is a point in 3D as shown in Fig. 12.20, STF codeword can be defined [9] as the collection of transmitted symbols within the parallelepiped, spanned by N_T transmit antennas, N_x OFDM symbol intervals, and N_c subcarriers. Thus, one STF codeword contains $N_T N_x N_c$ transmitted symbols

$$\{x_n^\mu(i), \mu = 1, \dots, N_T, n = 0, \dots, N_x-1, i = 0, \dots, N_c-1\}.$$

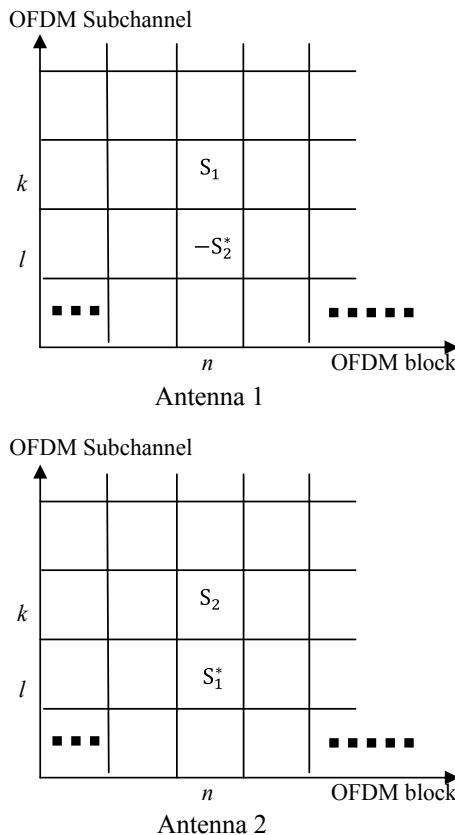


Fig. 12.19 SF coding

12.7 Problems

1. Consider Alamouti STBC with 2-transmit antennas. If the input bit stream is 11011110001001, determine the transmitted symbols from each antenna for each symbol interval with (i) QPSK modulation (ii) 16-QAM modulation.
2. A code matrix for STBC is given by

$$\begin{bmatrix} S_1 & S_2 & S_3 & S_4 \\ -S_2^* & S_1^* & -S_4^* & S_3^* \\ -S_3^* & -S_4^* & S_1^* & S_2^* \\ S_4 & -S_3 & -S_2 & S_1 \end{bmatrix}$$

- (i) Check for Orthogonality of the code
- (ii) Find the diversity order achieved by this code.

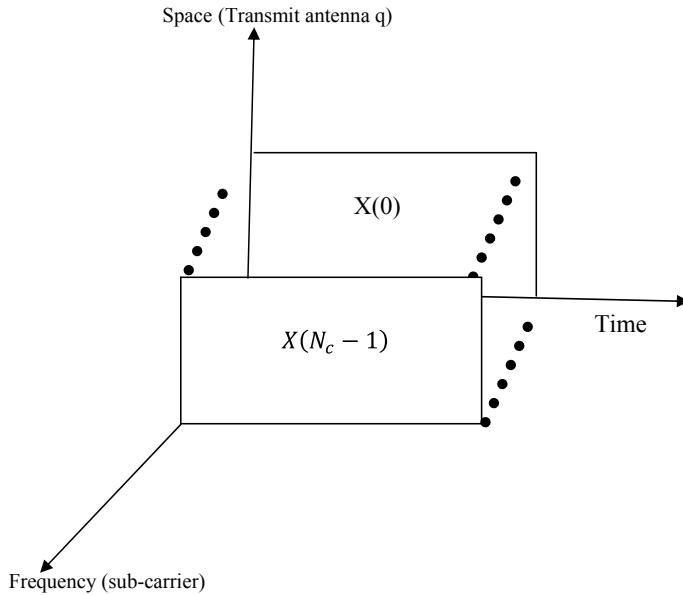


Fig. 12.20 STF coded OFDM

3. Consider a MIMO system with AWGN employing Alamouti STBC with two transmit and one receiving antennas. Determine the outage probabilities for the system
 - (i) When the channel is known at the receiver
 - (ii) When the channel is known at the transmitter.
4. Consider a 4-state QPSK STTC system. Determine the trellis-encoded symbol stream if the two input bit sequences are

$$\begin{bmatrix} I_t^1 \\ I_t^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & \dots \\ 0 & 1 & 1 & 0 & 1 & \dots \end{bmatrix}$$

5. Consider a 4-state QPSK STTC system. Determine the trellis-encoded symbol stream if the two input bit sequences are

$$\begin{bmatrix} I_t^1 \\ I_t^2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & \dots \\ 0 & 0 & 1 & 1 & 0 & \dots \end{bmatrix}$$

6. Consider a STTC 4-PSK system where the transmitted codeword is $C = 220313$, and a possible erroneous codeword is $c = 330122$. Determine the diversity gain of the system.
7. Consider the same data from the problem 11.5, determine the coding gain.

12.8 MATLAB Exercises

1. Write a MATLAB program to simulate the BER performance of 2-transmit 1-receive antenna Alamouti scheme using MMSE detection.
2. Write a MATLAB program to simulate the BER performance of 2-transmit 2-receive antenna Alamouti scheme using MMSE detection.
3. Write a MATLAB program to simulate the BER performance of MRC diversity technique with 4 receiving antennas and compare with the result of problem 2.
4. Write a MATLAB program to simulate the performance of 8-state QPSK STTC.
5. Write a MATLAB program to simulate the BER performance of STBC OFDM.

References

1. X.-B. Liang, Orthogonal designs with maximum rates. *IEEE Trans. Inf. Theory* **49**(10), 2468–2503 (2003)
2. V. Tarokh, N. Seshadri, A.R. Calderbank, Space-time codes for high data rate wireless communications: performance criterion and code construction. *IEEE Trans. Inf. Theory* **44**(2), 744–765 (1998)
3. V. Tarokh, A. Naguib, N. Seshadri, A.R. Calderbank, Space-time codes for high data rate wireless communication: performance criteria in the presence of channel estimation errors, mobility, and multiple paths. *IEEE Trans. Commun.* **47**(2), 199–207 (1999)
4. V. Tarokh, N. Seshadri, A.R. Calderbank, Space-time codes for wireless communication: code construction, in *IEEE 47th Vehicular Technology Conference*, vol. 2 (Phoenix, Arizona, 4–7 May 1997), pp. 637–641
5. S.M. Alamouti, A simple transmit diversity technique for wireless communications. *IEEE J. Sel. Areas Commun.* **16**(8), 1451–1458 (1998)
6. V. Tarokh, H. Jafarkhani, A. Calderbank, Space-time block coding for wireless communications: performance results. *IEEE J. Sel. Areas Commun.* **17**(3), 451–460 (1999)
7. V. Tarokh, H. Jafarkhani, A. Calderbank, Space-time block codes from orthogonal designs. *IEEE Trans. Inf. Theory* **45**(5), 1456–1467 (1999)
8. Z. Chent, J. Yuant, B. Vucetict, An improved space-time trellis coded modulation scheme on slow rayleigh fading channels, in *IEEE ICC*, pp. 1110–1116 (2001)
9. Z. Liu, Y. Xin, G.B. Giannakis, Space-time-frequency coded OFDM over frequency-selective fading channels. *IEEE Trans. Sig. Process.* **50**(10), 2465–2476 (2002)
10. K.F. Lee, D.B. Williams, A space-frequency transmitter diversity technique for OFDM systems, in *IEEE Global Communications Conference*, vol. 3 (2000), pp. 1473–1477 (27 Nov–1 Dec)
11. H. Jafarkhani, *Space-Time Coding Theory and Practice* (Cambridge University Press, 2005)
12. W. Zhang, X.-G. Xia, K.B. Letaief, Space-time/frequency coding for MIMO-OFDM in the next generation broadband wireless systems. *IEEE Wirel. Commun.* 32–43 (2007)

Chapter 13

Channel Codes Evolution for 5G



The next-generation mobile network must meet diversified demands. The International Telecommunication Union (ITU) has classified 5G mobile network services into three categories: enhanced mobile broadband (eMBB), ultra-reliable and low-latency communications (uRLLC), and massive machine type communications (mMTC). Channel codes play an important role in deciding the latency and reliability requirements of 5G communications.

This chapter covers the evolution of channel codes for the core services of 5G communications.

13.1 5G Requirements

5G system requirements are given in [1]. The data rate needs to increase by roughly 1000 times from 4G to 5G. The data traffic needs to increase by 10,000 times. The 5G edge rate (or 5% rate) ranges from 100 Mbps to as much as 1 Gbps.

The 5G needs to support a roundtrip latency of about 1 ms. More than 10–100 times higher number of connected devices are anticipated in 5G system. The energy efficiency (Joules per bit) needs to fall by at least 100 times. In addition, accurate positioning of the devices, strong level of security, smooth mobility between different types of radio access networks, and flexible and very dense deployments are required. There are other views of 5G requirements from industry and academia [1–5] as shown in Table 13.1.

It can be observed from Table 13.1, academia sets more aggressive targets and industry sets more realistic targets. However, their gap is not so big. They approximately achieve consensus on 5G requirements.

Among those 5G requirements, achieving high spectral efficiency is the key requirement because it provides us with much lower cost per bit. In order to achieve high spectral efficiency in 5G system, channel coding and modulation are the key part of the 5G physical layer. The channel coding and modulation of the 5G system

Table 13.1 5G requirements of industry and academia

	Samsung [2]	Nokia [3]	NTT docomo [4]	Metis [5]	Academia [1]
Data rate	50×	10×	100×	100×	1000×
Cell edge data rate	>1 Gbps	100 Mbps	–	–	>1 Gbps
Latency	<1 ms	<1 ms	<1 ms	<5×	<1 ms
Connected devices	10×	100×	100×	100×	100×
Energy efficiency	–	10 years M2M battery life	–	10× longer battery life	<100×
Cost efficiency	10×	–	–	–	–
Traffic	–	10,000×	–	>1000×	–

requires to achieve better spectral efficiency, support multiterminal encoding and decoding schemes in cooperative communications or heterogeneous networks, and reduce transmission and computation power.

In the following terms, 5G is distinct from 4G.

- High increased peak bit rate
- Larger data volume per unit area (i.e., high system spectral efficiency)
- High capacity to allow more devices connectivity concurrently and instantaneously
- Lower battery consumption
- Better connectivity irrespective of the geographic region
- Larger number of supporting devices
- Lower cost of infrastructural development
- Higher reliability of the communications

In 5G, the following three service categories are targeted.

- eMBB provides high data rates along with lower latency compared to 4G network. Services such as ULTRA HD and VR are main beneficiaries.
- In mMTC, devices in an area will be in very high numbers, should have support for long battery life, and have range more than conventional networks.
- In URLLC feature, latency is less than 1 ms (over the air), and targeted reliability is 99.99%.

13.2 QC-LDPC and Polar Codes for eMBB

3GPP has selected LDPC codes for data channel coding and polar codes for control channel coding for 5G enhanced mobile broadband (eMBB) service [6]. The data channel requirements in respect of information block lengths and code rates are different from the control channel requirements.

This section evaluates the performance of QPSK modulated QC-LDPC codes and polar codes for 5G eMBB service over AWGN channel using block error rate (BLER) as the performance measure.

13.2.1 Performance Evaluation of QC-LDPC Codes for eMBB Data Channel

The complete summary of utilized simulation parameters for data channel is tabulated in Table 13.2.

A uniform parity check matrix H is preferred for 5G eMBB channel coding [7]. The uniform base matrix means that code base matrix is derived from a uniform base matrix for any code block size or code rate. A subbase matrix of corresponding number of rows and columns is extracted from the uniform base matrix to support different code rates. The expanding factor (lift size) can be changed to support different code block sizes. For example, at rate (R) = 1/2, for $(W_{\max})/11n/$, the base parity check matrix H_b is constructed using $mb = 12$, $nb = 24$, and kb is to be chosen as 12. It is preferred [7] that the maximum information block size as K_{\max} is in the range of $6144 \leq K_{\max} \leq 8192$ and $kb = 20$ in most wireless wideband communication systems.

In this simulation, LDPC codes are generated using the base matrix given in [7] with permutation vector [1, 2, 3, 4, 5, 6, 7, 10, 11, 9, 8, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 0] and $zset = [4\ 6\ 8\ 10\ 12\ 14\ 16\ 20\ 24\ 28\ 32\ 40\ 48\ 56\ 64\ 80\ 96\ 112\ 128\ 160\ 192\ 224\ 256\ 320\ 384\ 448\ 512\ 640\ 768\ 896\ 1024]$ for the code rates and information block lengths shown in Table 13.1. The QPSK modulation is applied on each of the code block and white Gaussian noise of different powers is added to the modulated bits. The QPSK demodulation is performed on the noisy signal. The flooding belief propagation (BP) algorithm is applied on the demodulated signal for decoding to extract the information bits and comparing the extracted bits with the original K information bits, BLER is computed.

The BLER performance of QC-LDPC codes using QPSK modulation over AWGN for the code rates 1/3, 2/5, and 1/2 with information block lengths $K = 2000, 4000$, and 8000 is shown in Figs. 13.1, 13.2, and 13.3, respectively. The BLER values are computed by averaging over 100 independent trials.

Table 13.2 Simulation parameters

Channel	AWGN
Modulation	QPSK
Code rates	1/3, 2/5, 1/2
Code	QC-LDPC
Information block lengths	2000, 4000, 8000
Decoding mechanism	Flooding BP

Fig. 13.1 BLER performance of QC-LDPC codes with list decoding for $K = 2000$

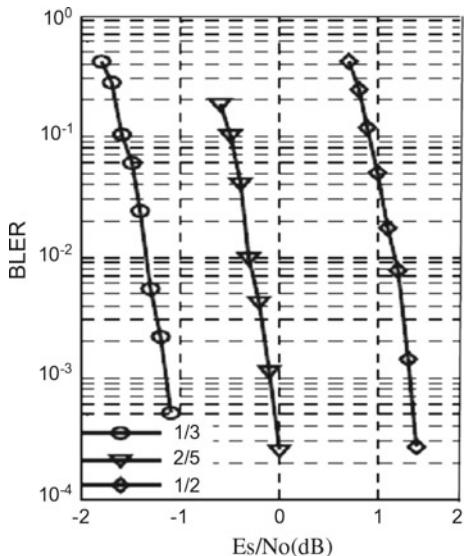
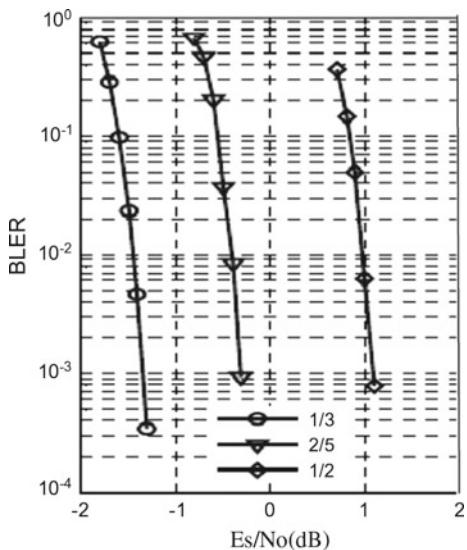


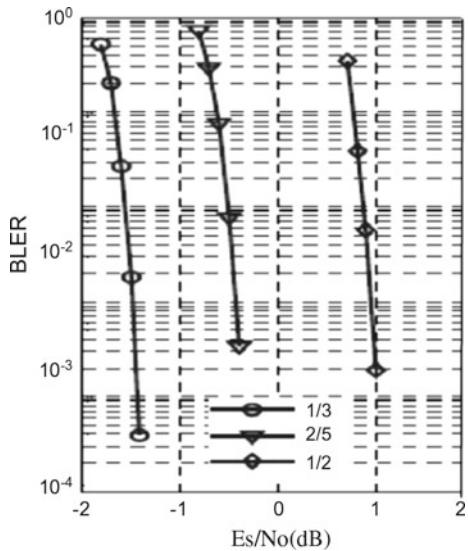
Fig. 13.2 BLER performance of QC-LDPC codes with list decoding for $K = 4000$



13.2.2 Performance Evaluation of Polar Codes for eMBB Control Channel

For downlink control channel, the information block size is expected [8] to be in the range of 80–100 bits on the higher end. Hence it is appropriate to use $N_{\max} = 256$ for DCI. This provides a mother code rate in the range of $R = 0.3\text{--}0.4$ for the

Fig. 13.3 BLER performance of QC-LDPC codes with list decoding for $K = 8000$



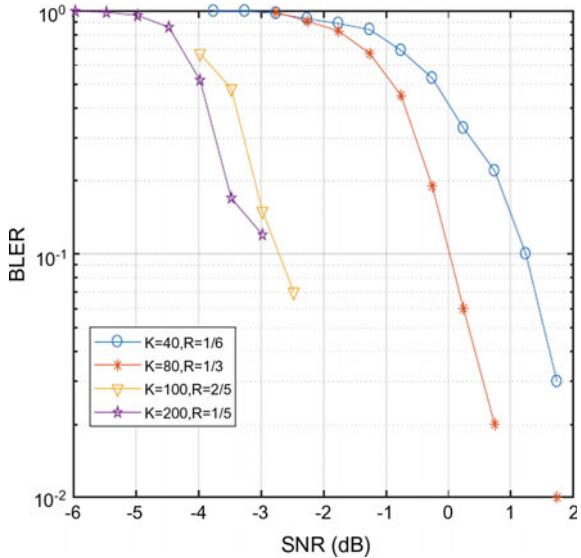
largest DCI. For smaller DCI, e.g., $K = 40$ bits, the mother code rate is $R = 0.16$. For uplink control channel, the information block size is expected to be in the range of 200 bits on the higher end. Hence, it is appropriate to use $N_{\max} = 1024$ for UCI. This provides a mother code rate in the range of $R = 0.2$ for the largest UCI. Since UCI does not require blind decoding of multiple candidates in one attempt, selection of N_{\max} can be more relaxed compared to DCI. The complete summary of utilized simulation parameters for data channel is tabulated in Table 13.3.

In this simulation, for each frame processed, cyclic redundancy code bits are appended to randomly generated K random information bits. The information bits with the appended bits are encoded to obtain a code of length N . The puncturing is performed on code length N to obtain the M bits to be transmitted. The M value is computed by dividing K by Rate R . The QPSK modulation is applied on each of the M bits and white Gaussian noise of different powers is added to the modulated bits.

Table 13.3 Simulation parameters

Channel	AWGN
Modulation	QPSK
Code rates	1/6, 1/3, 2/5 (downlink) 1/5 (uplink)
code	polar
Information block lengths	40, 80, 100 (down link) 200 (up link)
CRC length	19
Decoding mechanism	SCL
List size	8

Fig. 13.4 BLER performance for code rates $1/6, 1/3, 2/5, 1/5$ for $K = 40, 80, 100, 200$



The QPSK demodulation is performed on the noisy signal to obtain LLR values. The LLR values for the punctured bits are set to zero. The CRC aided SCL algorithm is applied on the padded LLR values for decoding. From the decoded bits of length $K + \text{CRCbitslength}$, the first K information bits are extracted and comparing these bits with the original K information bits, BLER is computed. The BLER performance of list decoding for CRC aided polar codes using QPSK modulation over AWGN for the code rates $1/6, 1/3, 2/5$, and $1/2$ with information block lengths $K = 40, 80, 100$, and 200 is shown in Fig. 13.4. The BLER values are computed by averaging over 100 independent trials.

13.3 Evaluation of Enhanced Turbo Codes and Polar Codes for URLLC

The complete summary of utilized simulation parameters are given in Table 13.4.

In Table 13.5, it may be noted that “1” means “non-punctured (transmitted) symbol” and “0” means “punctured (non-transmitted) symbol.”

The enhanced turbo codes are constructed using puncturing masks as tabulated in Table 13.5 and interleaver parameters as tabulated in Table 13.6.

The interleaving function for enhanced turbo codes is given by

$$\pi(i) = (Pi + S(i \bmod Q)) \bmod K$$

Table 13.4 Simulation parameters

Channel	AWGN
Modulation	QPSK
Code rates	1/3, 1/6, and 1/12
Information block lengths	32, 80, and 208 (without CRC)
CRC length	19
Decoding mechanism	SCL for Polar codes, and Max-log-MAP for enhanced turbo codes
List size	8
Iterations for enhanced turbo code	8

Table 13.5 Puncturing masks under URLLC scenario code rates

Rate (R)	Encoder	Puncturing mask
1/3	RSC (1, 15/13) ₈ , Feed forward polynomial (EF) = 15 Feedback polynomial (FB) = 13	No puncturing
1/6	RSC (1, 15/13, 17/13, 11/13) ₈ Feed forward polynomials (KF) = 15, 17, 11 Feedback polynomial (FB) = 13	Systematic 0000 Parity (FF = 15) 1111 Parity (FF = 17) 1111 Parity (FF = 11) 1111
1/12	RSC(1, 15/13, 17/13, 11/13, 16/13, 12/13, 17/13) ₈ Feed forward polynomials (FF)=15, 17, 11, 16, 12, 17 Feedback polynomial (FB) = 13	Systematic 0000 Parity (FF = 15) 1111 Parity (FF = 17) 1111 Parity (FF = 11) 1111 Parity (FF = 16) 1111 Parity (FF = 12) 1111 Parity (FF = 17) 1111

Table 13.6 Interleaver parameters

Q	P	K	$(S(0) \dots S(Q - 1))$
4	9	32	(3, 5, 3, 5)
4	11	80	(3, 63, 3, 63)
16	147	208	(8, 156, 31, 174, 10, 115, 98, 62, 152, 97, 16, 156, 37, 84, 112, 68)

where i stands for the data symbol address after interleaving denotes the corresponding address before interleaving.

The list decoding technique of [9] is used for decoding enhanced turbo codes in which a list Viterbi decoder [10] is applied to find a list containing the L most likely paths. The polar codes are constructed using design SNR independent method reported in [11] and decoded using successive cancellation list decoding given in [12].

The BLER performance of list decoding for CRC aided polar codes and list decoding for enhanced turbo codes using QPSK modulation over AWGN channel

for the code rates 1/3, 1/6, and 1/12 with information block length $K = 32$ are shown in Figs. 13.5, 13.6 and 13.7 provide the corresponding results for information block length $K = 80$ and 208 with code rates 1/3, 1/6, and 1/12, respectively [13]. The

Fig. 13.5 BLER performance of enhanced turbo and CRC aided polar codes with list decoding for $K = 32$, $R = 1/12, 1/6$, and $1/3$, and $L = 8$

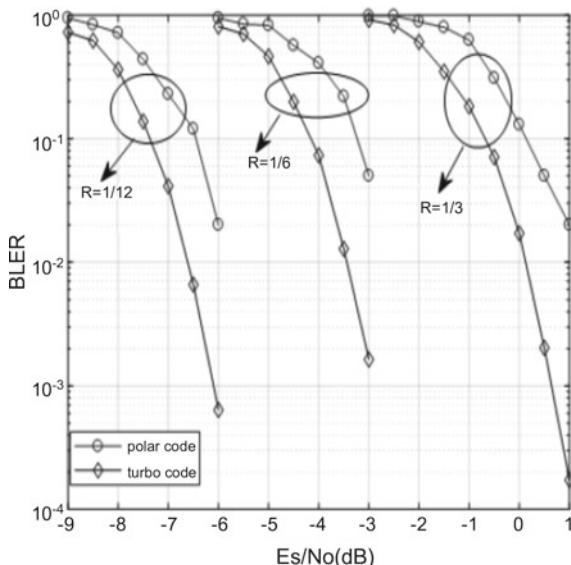
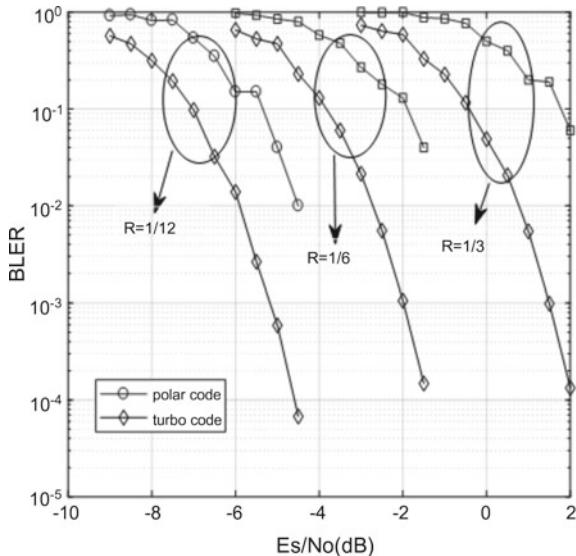
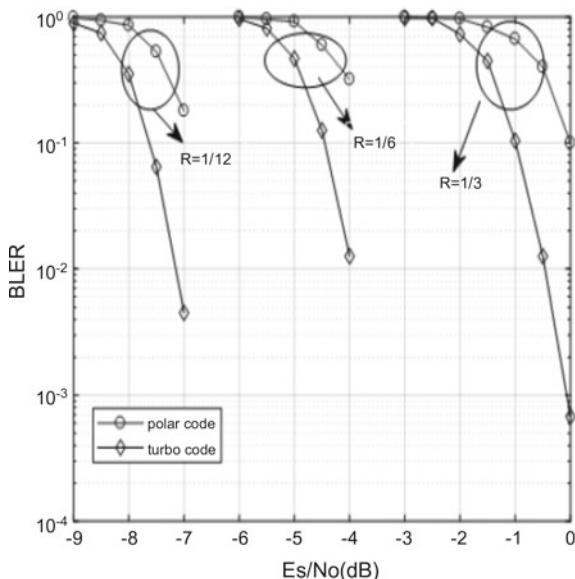


Fig. 13.6 BLER performance of enhanced turbo and CRC aided polar codes with list decoding for $K = 80$, $R = 1/12, 1/6$, and $1/3$, and $L = 8$

Fig. 13.7 BLER performance of enhanced turbo and CRC aided polar codes with list decoding for $K = 208$, $R = 1/12$, $1/6$, and $1/3$, and $L = 8$



BLER values are computed by averaging over 1000 independent trials for all the considered channels. From simulation results, it is observed that enhanced turbo codes with list decoding performs better than the CRC aided polar codes with list decoding for short information block lengths ($K = 32, 80$, and 208) at low code rates ($R = 1/3, 1/6$, and $1/12$) over AWGN channel with no error floor down to 10^{-3} to 10^{-4} under URLLC scenario. Further, it is observed that at low code rate BLER values are achieved at very low symbol energy to noise ratio (Es/No) values. But as the code rate increases, the same BLER values require higher Es/No (Fig. 13.5).

13.3.1 Decoding Latency

The ASIC using 2-radix processors leads to 3.2 ns decoding latency per information bit using 90 nm technology. Thus, the decoding latency is 3.2 μ s for maximum information block length of 1000 bits in URLLC. Further, lower latencies can be achieved by ASIC using radix 4 processors and shuffled turbo decoding in which lower and upper decoders are operated simultaneously. The quantified latency requirement reported in [14] for URLLC is 4.66 μ s. Hence, the URLLC latency requirement can be fulfilled by ASIC implementation of enhanced turbo list decoder. It is suggested in [14] that the strictest latency requirements of URLLC cannot be met by polar codes. However, latency reducing techniques without loss of BLER performance are existing [15] for SCL decoding of polar codes wherein multiple decision is made in decoding instead of single bit decision as done in [12].

13.3.2 Decoding Complexity

It is discussed in [16] that the computational complexity for all code rates can be kept constant by using the enhanced turbo decoder whereas the computational complexity of polar decoders increase at the low code rates under URLLC scenario.

13.4 Channel Codes for mMTC

The mMTC requirements are understandably quite different from the eMBB and URLLC usage scenarios. The key demands of the mMTC use case are mainly to design low complex/low-cost design which could operate for years while serving smaller throughput requirements. For many mMTC scenarios, the device might operate only with battery power and require to communicate over a longer period. Moreover, the cost of the device should be lower in order to deploy in massive numbers. In summary, we can identify these requirements as,

1. Good error performance with low throughput
2. Simple implementation
3. High energy efficiency

In general, NB-IOT also handles mMTC type of communications, where recently decided to use tail-biting convolutional codes (TBCC) as the coding scheme. To support backward compatibility for NB-IOT, the same coding scheme may be needed to use for mMTC scenario in 5G [17].

It is highly likely that mMTC will operate in lower modulation schemes with shorter information block sizes to satisfy low power requirements. Therefore, simulations are carried out on 20, 40, and 200 information block sizes with QPSK modulation scheme with simulation parameters as shown in Table 13.7.

The BLER performance of TBCC using QPSK modulation over AWGN channel for the code rates 1/3, 1/6, and 1/12 with information block length $K = 20, 40$, and 200 are shown in Figs. 13.8, 13.9, and 13.10, respectively.

Table 13.7 Simulation parameters

Channel	AWGN
Modulation	QPSK
Coding scheme	TBCC
Code rates	1/3, 1/6, and 1/12
Information block lengths	20, 40, and 200 (without CRC)
CRC length	8 for information block lengths 20 and 40 and 16 for information block length of 200
Decoding mechanism	Viterbi

Fig. 13.8 BLER performance of tail-biting convolutional codes for $K = 20$, $R = 1/12$, $1/6$, and $1/3$

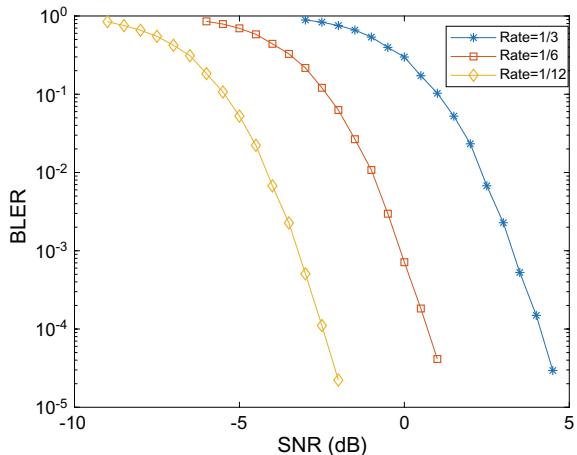


Fig. 13.9 BLER performance of tail-biting convolutional codes for $K = 40$, $R = 1/12$, $1/6$, and $1/3$

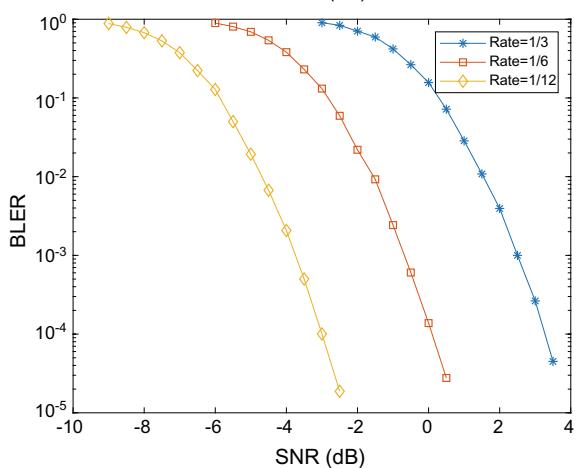
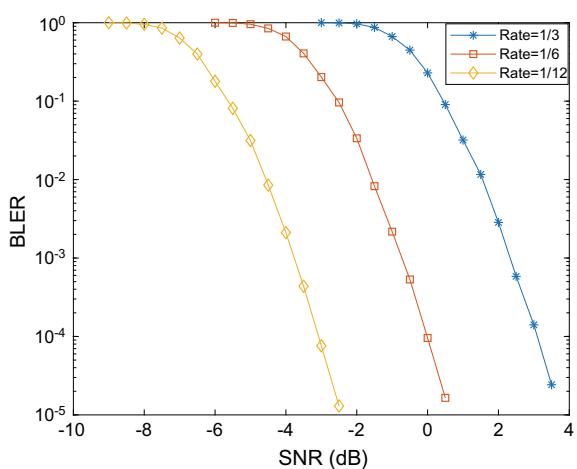


Fig. 13.10 BLER performance of tail-biting convolutional codes for $K = 200$, $R = 1/12$, $1/6$, and $1/3$



References

1. J.G. Andrews, S. Buzzi, W. Choi, S.V. Hanly, A. Lozano, A.C.K. Soong, J.C. Zhang, What will 5G be? *IEEE J. Sel. Areas Commun.* **32**(6), 1065–1082 (2014)
2. Samsung Electronics DMC R&D Center, “5G Vision,” Feb 2015
3. Nokia, 5G use cases and requirements. <https://networks.nokia.com>
4. NTT Docomo, 5G: Next-Generation Mobile Communications System (2014)
5. Metis Deliverable D1.1, Scenarios, requirements and KPIs for 5Gmobile and wireless system. Document Number: ICT-317669-METIS/D1.1
6. Final report of 3GPP TSG RAN WG1 87 v1.0.0
7. GPP TSG RAN WG1 AH NR Meeting, R1-1701473, ZTE, Spokane, USA, 16th–20th January 2017
8. GPP TSG RAN WG1 AH-NR Meeting R1-1700112, Ericsson, Spokane, USA, 16th–20th January 2017
9. A. Akmalikhodzhaev, A. Kozlov, New iterative turbo code list decoders. in Proceedings of International Symposium Problems of Redundancy in Information and Control Systems (2014)
10. N. Seshadri, C.E.W. Sundberg, List Viterbi decoding algorithms with applications. *IEEE Trans. Commun.* **42**(234), 313323 (1994)
11. R1-1611254, Details of the polar code design, 3GPP TSG RAN WG1 meeting, Huawei, HiSilicon, Nov 2016
12. A.B. Stimming, M.B. Parizi, A. Burg, LLR-based successive cancellation list decoding of polar codes. *IEEE Trans. Signal Process.* **63**(19), 5165–5179 (2015)
13. K. Deergha Rao, Performance analysis of enhanced turbo and polar codes with list decoding for URLLC in 5G systems, in Proceedings of IEEE International Conference for Convergence in Technology, 29–31 Mar 2019
14. Samsung, R1-1700981 Discussion on Channel Coding for URLLCChannel, in 3GPP TSG RAN WG1 AH NR, Jan 2017
15. B. Yuan, K.K. Parhi, Low-latency successive-cancellation list decoders for polar codes with multibit decision. *IEEE Trans. VLSI Syst.* **23**(10) (2015)
16. AccelerComm, R1-1612306 On the hardware implementation of channel decoders for short block lengths, in 3GPP TSG RAN WG1 87, Nov. 2016
17. R1-1609594, Performance of mMTC and URLLC channel coding candidates, Nokia, Alcatel-Lucent Shanghai Bell