

Error Correction Coding

Mathematical Methods and Algorithms

This Page Intentionally Left Blank

Error Correction Coding

This Page Intentionally Left Blank

Error Correction Coding

Mathematical Methods and Algorithms

Todd K. Moon

Utah State University



A JOHN WILEY & SONS, INC., PUBLICATION

Copyright © 2005 by John Wiley & Sons, Inc. All rights reserved.

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600, or on the web at www.copyright.com. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008.

Limit of Liability/Disclaimer of Warranty: While the publisher and author have used their best efforts in preparing this book, they make no representation or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not be suitable for your situation. You should consult with a professional where appropriate. Neither the publisher nor author shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

For general information on our other products and services please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993 or fax 317-572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print, however, may not be available in electronic format.

Library of Congress Cataloging-in-Publication Data:

Moon, Todd K.

Error correction coding : mathematical methods and algorithms / Todd K.

Moon.

p. cm.

Includes bibliographical references and index.

ISBN 0-471-64800-0 (cloth)

1. Engineering mathematics. 2. Error-correcting codes (Information theory)

I. Title.

TA331.M66 2005

621.382'0285'572—dc22

2004031019

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

Error Correction Coding

This Page Intentionally Left Blank

Preface

The purpose of this book is to provide a comprehensive introduction to error correction coding, including both classical block- and trellis-based codes and the recent developments in iteratively decoded codes such as turbo codes and low-density parity-check codes. The presentation is intended to provide a background useful both to engineers, who need to understand algorithmic aspects for the deployment and implementation of error correction coding, and to researchers, who need sufficient background to prepare them to read, understand, and ultimately contribute to the research literature. The practical algorithmic aspects are built upon a firm foundation of mathematics, which are carefully motivated and developed.

Pedagogical Features

Since its inception, coding theory has drawn from a rich and interacting variety of mathematical areas, including detection theory, information theory, linear algebra, finite geometries, combinatorics, optimization, system theory, probability, algebraic geometry, graph theory, statistical designs, Boolean functions, number theory, and modern algebra. The level of sophistication has increased over time: algebra has progressed from vector spaces to modules; practice has moved from polynomial interpolation to rational interpolation; Viterbi makes way for BCJR. This richness can be bewildering to students, particularly engineering students who are unaccustomed to posing problems and thinking abstractly. It is important, therefore, to motivate the mathematics carefully.

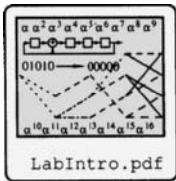
Some of the major pedagogical features of the book are as follows.

- While most engineering-oriented error-correction-coding textbooks clump the major mathematical concepts into a single chapter, in this book the concepts are developed over several chapters so they can be put to more immediate use. I have attempted to present the mathematics “just in time,” when they are needed and well-motivated. Groups and linear algebra suffice to describe linear block codes. Cyclic codes motivate polynomial rings. The design of cyclic codes motivates finite fields and associated number-theoretical tools. By interspersing the mathematical concepts with applications, a deeper and broader understanding is possible.
- For most engineering students, finite fields, the Berlekamp-Massey algorithm, the Viterbi algorithm, BCJR, and other aspects of coding theory are initially abstract and subtle. Software implementations of the algorithms brings these abstractions closer to a meaningful reality, bringing deeper understanding than is possible by simply working homework problems and taking tests. Even when students grasp the concepts well enough to do homework on paper, these programs provide a further emphasis, as well as tools to *help* with the homework. The understanding becomes *experiential*, more than merely conceptual.

Understanding of any subject typically improves when the student him- or herself has the chance to teach the material to someone (or something) else. A student must develop an especially clear understanding of a concept in order to “teach” it to something as dim-witted and literal-minded as a computer. In this process the

computer can provide feedback to the student through debugging and program testing that reinforces understanding.

In the coding courses I teach, students implement a variety of encoders and decoders, including Reed-Solomon encoders and decoders, convolutional encoders, turbo code decoders, and LDPC decoders. As a result of these programming activities, students move beyond an on-paper understanding, gaining a perspective of what coding theory can do and how to put it to work. A colleague of mine observed that many students emerge from a first course in coding theory more confused than informed. My experience with these programming exercises is that my students are, if anything, overconfident, and feel ready to take on a variety of challenges.



In this book, programming exercises are presented in a series of 13 Laboratory Exercises. These are supported with code providing most of the software “infrastructure,” allowing students to focus on the particular algorithm they are implementing.

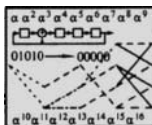
These labs also help with the coverage of the course material. In my course I am able to offload classroom instruction of some topics for students to read, with the assurance that the students will learn it solidly on their own as they implement it. (The Euclidean algorithm is one of these topics in my course.)

Research in error control coding can benefit from having a flexible library of tools for the computations, particularly since analytical results are frequently not available and simulations are required. The laboratory assignments presented here can form the foundation for a research library, with the added benefit that having written major components, the researcher can easily modify and extend them.

It is in light of these pedagogic features that this book bears the subtitle *Mathematical Methods and Algorithms*.

There is sufficient material in this book for a one- or two-semester course based on the book, even for instructors who prefer to focus less on implementational aspects and the laboratories.

Over 150 programs, functions and data files are associated with the text. The programs are written in Matlab,¹ C, or C++. Some of these include complete executables which provide “tables” of primitive polynomials (over any prime field), cyclotomic cosets and minimal polynomials, and BCH codes (not just narrow sense), avoiding the need to tabulate this material. Other functions include those used to make plots and compute results in the book. These provide example of how the theory is put into practice. Other functions include those used for the laboratory exercises. The files are highlighted in the book by the icon



as in the marginal note above. The files are available at the website

http://ftp.wiley.com/public/sci_tech_med/error_control

Other aspects of the book include the following:

¹Matlab is a registered trademark of The Mathworks, Inc.

- Many recent advances in coding have resulted from returning to the perspective of coding as a detection problem. Accordingly, the book starts off with a digital communication framework with a discussion of **detection theory**.
- Recent codes are capable of nearly achieving capacity. It is important, therefore, to understand what capacity is and what it means to transmit at capacity. Chapter 1 also summarizes **information theory**, to put coding into its historical and modern context. This information theory also is used in the EXIT chart analysis of turbo and LDPC codes.
- Pedagogically, Hamming codes are used to set the stage for the book by using them to demonstrate block codes, cyclic codes, trellises and Tanner graphs.
- Homework exercises are drawn from a variety of sources and are at a variety of levels. Some are numerical, testing basic understanding of concepts. Others provide the opportunity to prove or extend results from the text. Others extend concepts or provide new results. Because of the programming laboratories, exercises requiring decoding by hand of given bit sequences are few, since I am of the opinion that is better to know how to tell the computer than to do it by hand. I have drawn these exercises from a variety of sources, including problems that I faced as a student and those which I have given to students on homework and exams over the years.
- Number theoretic concepts such as divisibility, congruence, and the Chinese remainder theorem are developed.
- At points throughout the book, connections between the coding theoretic concepts and related topics are pointed out, such as **public key cryptography** and **shift register sequences**. These add spice and motivate students with the understanding that the tools they are learning have broad applicability.
- There has been considerable recent progress made in decoding Reed-Solomon codes by re-examining their original definition. Accordingly, Reed-Solomon codes are defined both in this primordial way (as the image of a polynomial function) and also using a generator polynomial having roots that are consecutive powers of a primitive element. This sets the stage for several decoding algorithms for Reed-Solomon codes, including frequency-domain algorithms, **Welch-Berlekamp algorithm** and the **soft-input Guruswami-Sudan algorithm**.
- **Turbo codes**, including EXIT chart analysis, are presented, with both BCJR and SOVA decoding algorithms. Both probabilistic and likelihood decoding viewpoints are presented.
- **LDPC codes** are presented with an emphasis on the decoding algorithm. Density evolution analysis is also presented.
- **Decoding algorithms on graphs** which subsume both turbo code and LDPC code decoders, are presented.
- A summary of **log likelihood algebra**, used in soft-decision decoding, is presented.
- **Space-time codes**, used for multi-antenna systems in fading channels, are presented.

Courses of Study

A variety of courses of study are possible. In the one-semester course I teach, I move quickly through principal topics of block, trellis, and iteratively-decoded codes. Here is an outline

of one possible one-semester course:

Chapter 1: Major topics only.

Chapter 2: All.

Chapter 3: Major topics.

Chapter 4: Most. Leave CRC codes and LFSR to labs.

Chapter 5: Most. Leave Euclidean algorithm to lab; skip CRT; skip RSA.

Chapter 6: Basic topics.

Chapter 12: Most. Skip puncturing, stack-oriented algorithms and trellis descriptions of block codes

Chapter 13: Most. Skip the V.34 material.

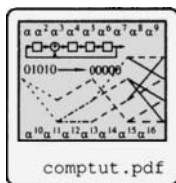
Chapter 14: Basic definition and the BCJR algorithm.

Chapter 15: Basic definition and the sum-product decoder.

A guide in selecting material for this course is: follow the labs. To get through all 13 labs, selectivity is necessary.

An alternative two-semester course could be a semester devoted to block codes followed by a semester on trellis and iteratively decoded codes. A two semester sequence could move straight through the book, with possible supplements from the research literature on topics of particular interest to the instructor.

The reader should be aware that theorems, lemmas, and corollaries are all numbered sequentially using the same counter in a chapter. Examples, definitions, figures, tables, and equations each have their own counters. Definitions, proofs and examples are all terminated by the symbol \square .



Use of Computers

The computer-based labs provide a means of working out some of the computational details that otherwise might require drudgery. There are in addition many tools available, both for modest cost and for free. The brief tutorial `comptut.pdf` provides an introduction to `gap` and `magma`, both of which can be helpful to students doing homework or research in this area.

Acknowledgments

In my mind, the purposes of a textbook are these:

1. To provide a topographical map into the field of study, showing the peaks and the important roads to them. (However, in an area as rich as coding theory, it is unfortunately impossible to be exhaustive.)
2. To provide specific details about important techniques.
3. To present challenging exercises that will strengthen students' understanding of the material and present new perspectives.
4. To have some reference value, so that practitioners can continue to use it.

5. To provide references to literature that will lead readers to make their own discoveries. (With a rapidly-changing field, the references can only provide highlights; web-based searches have changed the nature of the game. Nevertheless, having a starting point in the literature is still important.)

A significant difficulty I have faced is selection. The terrain is so richly textured that it cannot be mapped out in a single book. Every conference and every issue of the *IEEE Transactions on Information Theory* yields new and significant results. Publishing restrictions and practicality limit this book from being encyclopedic. My role as author has been merely to select what parts of the map to include and to present them in a pedagogically useful way. In so doing, I have aimed to choose tools for the general practitioner and student. Other than that selective role, no claim of creation is intended; I hope I have given credit as appropriate where it is due.

This book is a result of teaching a course in error correction coding at Utah State University for over a decade. Over that time, I have taught out of the books [33], [373], and [203], and my debt to these books is clear. Parts of some chapters grew out of lecture notes based on these books and the connections will be obvious. I have felt compelled to include many of the exercises from the first coding course I took out of [203]. These books have defined for me the *sine qua non* of error-correction coding texts. I am also indebted to [220] for its rich theoretical treatment, [303] for presentation of trellis coding material, [350] for discussion of bounds, [141] for exhaustive treatment of turbo coding methods, and to the many great researchers and outstanding expositors whose works have contributed to my understanding.

I am grateful for the supportive environment at Utah State University that has made it possible to undertake and to complete this task. Students in coding classes over the years have contributed to this material, and the students in ECE 7670 class of Spring 2005 have combed carefully through the text. Stewart Weber and Ray Rallison have improved my C++ code. Thanks to Ojas Chauhan, who produced the performance curves for convolutional codes. I am especially grateful to John Crockett, who gave a particularly careful reading and contributed to the EXIT charts for LDPC codes. He also did the solutions for the first three chapters of the solutions manual. With all the help I have had in trying to produce clean copy, I alone am responsible for any remaining errors.

To my six wonderful children — Leslie, Kyra, Kaylie, Jennie, Kiana, and Spencer — and my wife Barbara, who have seen me slip away too often and too long to write, I express my deep gratitude for their trust and patience. In the end, all I do is for them.

T.K.M
Logan, UT, Mar. 2005

This Page Intentionally Left Blank

Contents

Preface	vii
List of Program Files	xxxi
List of Laboratory Exercises	xxxii
List of Algorithms	xxxiv
List of Figures	xl
List of Tables	xlii
List of Boxes	xliii
Part I Introduction and Foundations	1
1 A Context for Error Correction Coding	2
1.1 Purpose of This Book	2
1.2 Introduction: Where Are Codes?	2
1.3 The Communications System	4
1.4 Basic Digital Communications	9
1.4.1 Binary Phase-Shift Keying	10
1.4.2 More General Digital Modulation	11
1.5 Signal Detection	14
1.5.1 The Gaussian Channel	14
1.5.2 MAP and ML Detection	16
1.5.3 Special Case: Binary Detection	18
1.5.4 Probability of Error for Binary Detection	19
1.5.5 Bounds on Performance: The Union Bound	22
1.5.6 The Binary Symmetric Channel	23
1.5.7 The BSC and the Gaussian Channel Model	25
1.6 Memoryless Channels	25
1.7 Simulation and Energy Considerations for Coded Signals	26
1.8 Some Important Definitions	27
1.8.1 Detection of Repetition Codes Over a BSC	28
1.8.2 Soft-Decision Decoding of Repetition Codes Over the AWGN	32
1.8.3 Simulation of Results	33
1.8.4 Summary	33
1.9 Hamming Codes	34
1.9.1 Hard-Input Decoding Hamming Codes	35
1.9.2 Other Representations of the Hamming Code	36
An Algebraic Representation	37
A Polynomial Representation	37

	A Trellis Representation	38
	The Tanner Graph Representation	38
1.10	The Basic Questions	39
1.11	Historical Milestones of Coding Theory	40
1.12	A Bit of Information Theory	40
1.12.1	Definitions for Discrete Random Variables	40
	Entropy and Conditional Entropy	40
	Relative Entropy, Mutual Information, and Channel Capacity	41
1.12.2	Definitions for Continuous Random Variables	43
1.12.3	The Channel Coding Theorem	45
1.12.4	“Proof” of the Channel Coding Theorem	45
1.12.5	Capacity for the Continuous-Time AWGN Channel	49
1.12.6	Transmission at Capacity with Errors	51
1.12.7	The Implication of the Channel Coding Theorem	52
Lab 1	Simulating a Communications Channel	53
	Objective	53
	Background	53
	Use of Coding in Conjunction with the BSC	53
	Assignment	54
	Programming Part	54
	Resources and Implementation Suggestions	54
1.13	Exercises	56
1.14	References	60
 Part II Block Codes		61
2	Groups and Vector Spaces	62
2.1	Introduction	62
2.2	Groups	62
2.2.1	Subgroups	65
2.2.2	Cyclic Groups and the Order of an Element	66
2.2.3	Cosets	67
2.2.4	Lagrange’s Theorem	68
2.2.5	Induced Operations; Isomorphism	69
2.2.6	Homomorphism	72
2.3	Fields: A Prelude	73
2.4	Review of Linear Algebra	75
2.5	Exercises	80
2.6	References	82
3	Linear Block Codes	83
3.1	Basic Definitions	83
3.2	The Generator Matrix Description of Linear Block Codes	84
3.2.1	Rudimentary Implementation	86
3.3	The Parity Check Matrix and Dual Codes	86
3.3.1	Some Simple Bounds on Block Codes	88
3.4	Error Detection and Correction over Hard-Input Channels	90

3.4.1	Error Detection	90
3.4.2	Error Correction: The Standard Array	90
3.5	Weight Distributions of Codes and Their Duals	95
3.6	Hamming Codes and Their Duals	97
3.7	Performance of Linear Codes	98
3.7.1	Error detection performance	99
3.7.2	Error Correction Performance	100
3.7.3	Performance for Soft-Decision Decoding	103
3.8	Erasur e Decoding	104
3.8.1	Binary Erasure Decoding	105
3.9	Modifications to Linear Codes	105
3.10	Best Known Linear Block Codes	107
3.11	Exercises	107
3.12	References	112
4	Cyclic Codes, Rings, and Polynomials	113
4.1	Introduction	113
4.2	Basic Definitions	113
4.3	Rings	114
4.3.1	Rings of Polynomials	115
4.4	Quotient Rings	116
4.5	Ideals in Rings	118
4.6	Algebraic Description of Cyclic Codes	120
4.7	Nonsystematic Encoding and Parity Check	122
4.8	Systematic Encoding	124
4.9	Some Hardware Background	126
4.9.1	Computational Building Blocks	126
4.9.2	Sequences and Power series	127
4.9.3	Polynomial Multiplication	128
Last-Element-First Processing	128	
First-Element-First Processing	128	
4.9.4	Polynomial division	129
Last-Element-First Processing	129	
4.9.5	Simultaneous Polynomial Division and Multiplication	132
First-Element-First Processing	132	
4.10	Cyclic Encoding	133
4.11	Syndrome Decoding	137
4.12	Shortened Cyclic Codes	143
Method 1: Simulating the Extra Clock Shifts	144	
Method 2: Changing the Error Pattern Detection Circuit	147	
4.13	Binary CRC Codes	147
4.13.1	Byte-Oriented Encoding and Decoding Algorithms	150
4.13.2	CRC Protecting Data Files or Data Packets	153
Appendix 4.A	Linear Feedback Shift Registers	154
Appendix 4.A.1	Basic Concepts	154
Appendix 4.A.2	Connection With Polynomial Division	157
Appendix 4.A.3	Some Algebraic Properties of Shift Sequences	160

Lab 2 Polynomial Division and Linear Feedback Shift Registers . . .	161
Objective	161
Preliminary Exercises	161
Programming Part: BinLFSR	161
Resources and Implementation Suggestions	161
Programming Part: BinPolyDiv	162
Follow-On Ideas and Problems	162
Lab 3 CRC Encoding and Decoding	162
Objective	163
Preliminary	163
Programming Part	163
Resources and Implementation Suggestions	163
4.14 Exercises	165
4.15 References	170
5 Rudiments of Number Theory and Algebra	171
5.1 Motivation	171
5.2 Number Theoretic Preliminaries	175
5.2.1 Divisibility	175
5.2.2 The Euclidean Algorithm and Euclidean Domains	177
5.2.3 The Sugiyama Algorithm	182
5.2.4 Congruence	184
5.2.5 The ϕ Function	185
5.2.6 Some Cryptographic Payoff	186
Fermat's Little Theorem	186
RSA Encryption	187
5.3 The Chinese Remainder Theorem	188
5.3.1 The CRT and Interpolation	190
The Evaluation Homomorphism	190
The Interpolation Problem	191
5.4 Fields	193
5.4.1 An Examination of \mathbb{R} and \mathbb{C}	194
5.4.2 Galois Field Construction: An Example	196
5.4.3 Connection with Linear Feedback Shift Registers	199
5.5 Galois Fields: Mathematical Facts	200
5.6 Implementing Galois Field Arithmetic	204
5.6.1 Zech Logarithms	204
5.6.2 Hardware Implementations	205
5.7 Subfields of Galois Fields	206
5.8 Irreducible and Primitive polynomials	207
5.9 Conjugate Elements and Minimal Polynomials	209
5.9.1 Minimal Polynomials	212
5.10 Factoring $x^n - 1$	215
5.11 Cyclotomic Cosets	217
Appendix 5.A How Many Irreducible Polynomials Are There?	218
Appendix 5.A.1 Solving for I_m Explicitly: The Moebius Function	222
Lab 4 Programming the Euclidean Algorithm	223

Objective	223
Preliminary Exercises	223
Background	223
Programming Part	223
Lab 5 Programming Galois Field Arithmetic	224
Objective	224
Preliminary Exercises	224
Programming Part	224
5.12 Exercises	225
5.13 References	234
6 BCH and Reed-Solomon Codes: Designer Cyclic Codes	235
6.1 BCH Codes	235
6.1.1 Designing BCH Codes	235
6.1.2 The BCH Bound	237
6.1.3 Weight Distributions for Some Binary BCH Codes	239
6.1.4 Asymptotic Results for BCH Codes	240
6.2 Reed-Solomon Codes	242
6.2.1 Reed-Solomon Construction 1	242
6.2.2 Reed-Solomon Construction 2	243
6.2.3 Encoding Reed-Solomon Codes	244
6.2.4 MDS Codes and Weight Distributions for RS Codes	245
6.3 Decoding BCH and RS Codes: The General Outline	247
6.3.1 Computation of the Syndrome	247
6.3.2 The Error Locator Polynomial	248
6.3.3 Chien Search	248
6.4 Finding the Error Locator Polynomial	250
6.4.1 Simplifications for Binary Codes and Peterson's Algorithm	251
6.4.2 Berlekamp-Massey Algorithm	253
6.4.3 Characterization of LFSR Length in Massey's Algorithm	255
6.4.4 Simplifications for Binary Codes	259
6.5 Non-Binary BCH and RS Decoding	261
6.5.1 Forney's Algorithm	262
6.6 Euclidean Algorithm for the Error Locator Polynomial	266
6.7 Erasure Decoding for Nonbinary BCH or RS codes	267
6.8 Galois Field Fourier Transform Methods	269
6.8.1 Equivalence of the Two Reed-Solomon Code Constructions	274
6.8.2 Frequency-Domain Decoding	275
6.9 Variations and Extensions of Reed-Solomon Codes	276
6.9.1 Simple Modifications	276
6.9.2 Generalized Reed-Solomon Codes and Alternant Codes	277
6.9.3 Goppa Codes	278
6.9.4 Decoding Alternant Codes	280
6.9.5 The McEliece Public Key Cryptosystem	280
Lab 6 Programming the Berlekamp-Massey Algorithm	281
Background	281
Assignment	281

Preliminary Exercises	281
Programming Part	281
Resources and Implementation Suggestions	282
Lab 7 Programming the BCH Decoder	283
Objective	283
Preliminary Exercises	283
Programming Part	283
Resources and Implementation Suggestions	283
Follow-On Ideas and Problems	284
Lab 8 Reed-Solomon Encoding and Decoding	284
Objective	284
Background	284
Programming Part	284
Appendix 6.A Proof of Newton's Identities	285
6.10 Exercises	287
6.11 References	291
7 Alternate Decoding Algorithms for Reed-Solomon Codes	293
7.1 Introduction: Workload for Reed-Solomon Decoding	293
7.2 Derivations of Welch-Berlekamp Key Equation	293
7.2.1 The Welch-Berlekamp Derivation of the WB Key Equation	294
7.2.2 Derivation From the Conventional Key Equation	298
7.3 Finding the Error Values	300
7.4 Methods of Solving the WB Key Equation	302
7.4.1 Background: Modules	302
7.4.2 The Welch-Berlekamp Algorithm	303
7.4.3 Modular Solution of the WB Key Equation	310
7.5 Erasure Decoding with the Welch-Berlekamp Key Equation	321
7.6 The Guruswami-Sudan Decoding Algorithm and Soft RS Decoding	322
7.6.1 Bounded Distance, ML, and List Decoding	322
7.6.2 Error Correction by Interpolation	323
7.6.3 Polynomials in Two Variables	324
Degree and Monomial Order	325
Zeros and Multiple Zeros	328
7.6.4 The GS Decoder: The Main Theorems	330
The Interpolation Theorem	331
The Factorization Theorem	331
The Correction Distance	333
The Number of Polynomials in the Decoding List	335
7.6.5 Algorithms for Computing the Interpolation Step	337
Finding Linearly Dependent Columns: The Feng-Tzeng Algorithm	338
Finding the Intersection of Kernels: The Kötter Algorithm	342
7.6.6 A Special Case: $m = 1$ and $L = 1$	348
7.6.7 The Roth-Ruckenstein Algorithm	350
What to Do with Lists of Factors?	354
7.6.8 Soft-Decision Decoding of Reed-Solomon Codes	358
Notation	358

	A Factorization Theorem	360
	Mapping from Reliability to Multiplicity	361
	The Geometry of the Decoding Regions	363
	Computing the Reliability Matrix	364
7.7	Exercises	365
7.8	References	368
8	Other Important Block Codes	369
8.1	Introduction	369
8.2	Hadamard Matrices, Codes, and Transforms	369
8.2.1	Introduction to Hadamard Matrices	369
8.2.2	The Paley Construction of Hadamard Matrices	371
8.2.3	Hadamard Codes	374
8.3	Reed-Muller Codes	375
8.3.1	Boolean Functions	375
8.3.2	Definition of the Reed-Muller Codes	376
8.3.3	Encoding and Decoding Algorithms for First-Order RM Codes	379
	Encoding $RM(1, m)$ Codes	379
	Decoding $RM(1, m)$ Codes	379
	Expediting Decoding Using the Fast Hadamard Transform	382
8.3.4	The Reed Decoding Algorithm for $RM(r, m)$ Codes, $r \geq 1$	384
	Details for an $RM(2, 4)$ Code	384
	A Geometric Viewpoint	387
8.3.5	Other Constructions of Reed-Muller Codes	391
8.4	Building Long Codes from Short Codes: The Squaring Construction	392
8.5	Quadratic Residue Codes	396
8.6	Golay Codes	398
8.6.1	Decoding the Golay Code	400
	Algebraic Decoding of the \mathcal{G}_{23} Golay Code	400
	Arithmetic Decoding of the \mathcal{G}_{24} Code	401
8.7	Exercises	403
8.8	References	404
9	Bounds on Codes	406
9.1	The Gilbert-Varshamov Bound	409
9.2	The Plotkin Bound	410
9.3	The Griesmer Bound	411
9.4	The Linear Programming and Related Bounds	413
9.4.1	Krawtchouk Polynomials	415
9.4.2	Character	415
9.4.3	Krawtchouk Polynomials and Characters	416
9.5	The McEliece-Rodemich-Rumsey-Welch Bound	418
9.6	Exercises	420
9.7	References	424

10 Bursty Channels, Interleavers, and Concatenation	425
10.1 Introduction to Bursty Channels	425
10.2 Interleavers	425
10.3 An Application of Interleaved RS Codes: Compact Discs	427
10.4 Product Codes	430
10.5 Reed-Solomon Codes	431
10.6 Concatenated Codes	432
10.7 Fire Codes	433
10.7.1 Fire Code Definition	433
10.7.2 Decoding Fire Codes: Error Trapping Decoding	435
10.8 Exercises	437
10.9 References	438
11 Soft-Decision Decoding Algorithms	439
11.1 Introduction and General Notation	439
11.2 Generalized Minimum Distance Decoding	441
11.2.1 Distance Measures and Properties	442
11.3 The Chase Decoding Algorithms	445
11.4 Halting the Search: An Optimality Condition	445
11.5 Ordered Statistic Decoding	447
11.6 Exercises	449
11.7 References	450
Part III Codes on Graphs	451
12 Convolutional Codes	452
12.1 Introduction and Basic Notation	452
12.1.1 The State	456
12.2 Definition of Codes and Equivalent Codes	458
12.2.1 Catastrophic Encoders	461
12.2.2 Polynomial and Rational Encoders	464
12.2.3 Constraint Length and Minimal Encoders	465
12.2.4 Systematic Encoders	468
12.3 Decoding Convolutional Codes	469
12.3.1 Introduction and Notation	469
12.3.2 The Viterbi Algorithm	471
12.3.3 Some Implementation Issues	481
The Basic Operation: Add-Compare-Select	481
Decoding Streams of Data: Windows on the Trellis	481
Output Decisions	482
Hard and Soft Decoding; Quantization	484
Synchronization Issues	486
12.4 Some Performance Results	487
12.5 Error Analysis for Convolutional Codes	491
12.5.1 Enumerating Paths Through the Trellis	493
Enumerating on More Complicated Graphs: Mason's Rule	496

12.5.2	Characterizing the Node Error Probability P_e and the Bit Error Rate P_b	498
12.5.3	A Bound on P_d for Discrete Channels	501
	Performance Bound on the BSC	503
12.5.4	A Bound on P_d for BPSK Signaling Over the AWGN Channel	503
12.5.5	Asymptotic Coding Gain	504
12.6	Tables of Good Codes	505
12.7	Puncturing	507
12.7.1	Puncturing to Achieve Variable Rate	509
12.8	Suboptimal Decoding Algorithms for Convolutional Codes	510
12.8.1	Tree Representations	511
12.8.2	The Fano Metric	511
12.8.3	The Stack Algorithm	515
12.8.4	The Fano Algorithm	517
12.8.5	Other Issues for Sequential Decoding	520
12.8.6	A Variation on the Viterbi Algorithm: The M Algorithm	521
12.9	Convolutional Codes as Block Codes	522
12.10	Trellis Representations of Block and Cyclic Codes	523
12.10.1	Block Codes	523
12.10.2	Cyclic Codes	524
12.10.3	Trellis Decoding of Block Codes	525
Lab 9	Programming Convolutional Encoders	526
	Objective	526
	Background	526
	Programming Part	526
Lab 10	Convolutional Decoders: The Viterbi Algorithm	528
	Objective	528
	Background	528
	Programming Part	528
12.11	Exercises	529
12.12	References	533
13	Trellis Coded Modulation	535
13.1	Adding Redundancy by Adding Signals	535
13.2	Background on Signal Constellations	535
13.3	TCM Example	537
13.3.1	The General Ungerboeck Coding Framework	544
13.3.2	The Set Partitioning Idea	545
13.4	Some Error Analysis for TCM Codes	546
13.4.1	General Considerations	546
13.4.2	A Description of the Error Events	548
13.4.3	Known Good TCM Codes	552
13.5	Decoding TCM Codes	554
13.6	Rotational Invariance	556
	Differential Encoding	558
	Constellation Labels and Partitions	559
13.7	Multidimensional TCM	561

13.7.1	Some Advantages of Multidimensional TCM	562
13.7.2	Lattices and Sublattices	563
	Basic Definitions	563
	Common Lattices	565
	Sublattices and Cosets	566
	The Lattice Code Idea	567
	Sources of Coding Gain in Lattice Codes	567
	Some Good Lattice Codes	571
13.8	The V.34 Modem Standard	571
	Lab 11 Trellis-Coded Modulation Encoding and Decoding	578
	Objective	578
	Background	578
	Programming Part	578
13.9	Exercises	578
13.10	References	580

Part IV Iteratively Decoded Codes 581

14 Turbo Codes	582	
14.1	Introduction	582
14.2	Encoding Parallel Concatenated Codes	584
14.3	Turbo Decoding Algorithms	586
14.3.1	The MAP Decoding Algorithm	588
14.3.2	Notation	588
14.3.3	Posterior Probability	590
14.3.4	Computing α_t and β_t	592
14.3.5	Computing γ_t	593
14.3.6	Normalization	594
14.3.7	Summary of the BCJR Algorithm	596
14.3.8	A Matrix/Vector Formulation	597
14.3.9	Comparison of the Viterbi and BCJR Algorithms	598
14.3.10	The BCJR Algorithm for Systematic Codes	598
14.3.11	Turbo Decoding Using the BCJR Algorithm	600
	The Terminal State of the Encoders	602
14.3.12	Likelihood Ratio Decoding	602
	Log Prior Ratio $\lambda_{p,t}$	603
	Log Posterior $\lambda_{s,t}^{(0)}$	605
14.3.13	Statement of the Turbo Decoding Algorithm	605
14.3.14	Turbo Decoding Stopping Criteria	605
	The Cross Entropy Stopping Criterion	606
	The Sign Change Ratio (SCR) Criterion	607
	The Hard Decision Aided (HDA) Criterion	608
14.3.15	Modifications of the MAP Algorithm	608
	The Max-Log-MAP Algorithm	608
14.3.16	Corrections to the Max-Log-MAP Algorithm	609
14.3.17	The Soft Output Viterbi Algorithm	610
14.4	On the Error Floor and Weight Distributions	612

14.4.1	The Error Floor	612
14.4.2	Spectral Thinning and Random Interleavers	614
14.4.3	On Interleavers	618
14.5	EXIT Chart Analysis	619
14.5.1	The EXIT Chart	622
14.6	Block Turbo Coding	623
14.7	Turbo Equalization	626
14.7.1	Introduction to Turbo Equalization	626
14.7.2	The Framework for Turbo Equalization	627
Lab 12	Turbo Code Decoding	629
	Objective	629
	Background	629
	Programming Part	629
14.8	Exercises	629
14.9	References	632
15	Low-Density Parity-Check Codes	634
15.1	Introduction	634
15.2	LDPC Codes: Construction and Notation	635
15.3	Tanner Graphs	638
15.4	Transmission Through a Gaussian Channel	638
15.5	Decoding LDPC Codes	640
15.5.1	The Vertical Step: Updating $q_{mn}(x)$	641
15.5.2	Horizontal Step: Updating $r_{mn}(x)$	644
15.5.3	Terminating and Initializing the Decoding Algorithm	647
15.5.4	Summary of the Algorithm	648
15.5.5	Message Passing Viewpoint	649
15.5.6	Likelihood Ratio Decoder Formulation	649
15.6	Why Low-Density Parity-Check Codes?	653
15.7	The Iterative Decoder on General Block Codes	654
15.8	Density Evolution	655
15.9	EXIT Charts for LDPC Codes	659
15.10	Irregular LDPC Codes	660
15.10.1	Degree Distribution Pairs	662
15.10.2	Some Good Codes	664
15.10.3	Density Evolution for Irregular Codes	664
15.10.4	Computation and Optimization of Density Evolution	667
15.10.5	Using Irregular Codes	668
15.11	More on LDPC Code Construction	668
15.11.1	A Construction Based on Finite Geometries	668
15.11.2	Constructions Based on Other Combinatoric Objects	669
15.12	Encoding LDPC Codes	669
15.13	A Variation: Low-Density Generator Matrix Codes	671
15.14	Serial Concatenated Codes; Repeat-Accumulate Codes	671
15.14.1	Irregular RA Codes	673
Lab 13	Programming an LDPC Decoder	674
	Objective	674

Background	674
Assignment	675
Numerical Considerations	675
15.15 Exercises	676
15.16 References	679
16 Decoding Algorithms on Graphs	680
16.1 Introduction	680
16.2 Operations in Semirings	681
16.3 Functions on Local Domains	681
16.4 Factor Graphs and Marginalization	686
16.4.1 Marginalizing on a Single Variable	687
16.4.2 Marginalizing on All Individual Variables	691
16.5 Applications to Coding	694
16.5.1 Block Codes	694
16.5.2 Modifications to Message Passing for Binary Variables	695
16.5.3 Trellis Processing and the Forward/Backward Algorithm	696
16.5.4 Turbo Codes	699
16.6 Summary of Decoding Algorithms on Graphs	699
16.7 Transformations of Factor Graphs	700
16.7.1 Clustering	700
16.7.2 Stretching Variable Nodes	701
16.7.3 Exact Computation of Graphs with Cycles	702
16.8 Exercises	706
16.9 References	708
Part V Space-Time Coding	709
17 Fading Channels and Space-Time Codes	710
17.1 Introduction	710
17.2 Fading Channels	710
17.2.1 Rayleigh Fading	712
17.3 Diversity Transmission and Reception: The MIMO Channel	714
17.3.1 The Narrowband MIMO Channel	716
17.3.2 Diversity Performance with Maximal-Ratio Combining	717
17.4 Space-Time Block Codes	719
17.4.1 The Alamouti Code	719
17.4.2 A More General Formulation	721
17.4.3 Performance Calculation	721
Real Orthogonal Designs	723
Encoding and Decoding Based on Orthogonal Designs	724
Generalized Real Orthogonal Designs	726
17.4.4 Complex Orthogonal Designs	727
Future Work	728
17.5 Space-Time Trellis Codes	728
17.5.1 Concatenation	729
17.6 How Many Antennas?	732

17.7 Estimating Channel Information	733
17.8 Exercises	733
17.9 References	734
A Log Likelihood Algebra	735
A.1 Exercises	737
References	739
Index	750

This Page Intentionally Left Blank

List of Program Files

comptut.pdf	Introduction to gap and magma	x
qf.c	Q function	20
qf.m	Q function	20
bpskprobplot.m	Set up to plot prob. of error for BPSK	21
bpskprob.m	Plot probability of error for BPSK	21
recodeprob.m	Compute error probability for $(n, 1)$ repetition codes	32
testrecode.cc	Test the repetition code performance	33
repcodes.m	Plot results for repetition code	33
mindist.m	Find minimum distance using exhaustive search	34
hamcode74pe.m	Probability of error for $(7, 4)$ Hamming code	36
nchoosektest.m	Compute $\binom{n}{k}$ and test for $k < 0$	36
plotcapcmp.m	Capacity of the AWGN channel and BAWGNC channel	45
cawgnc2.m	Compute capacity of AWGN channel	45
cbawgnc2.m	Compute the capacity of the BAWGN channel	45
h2.m	Compute the binary entropy function	45
plotcbawn2.m	Plot capacity for AWGN and BAWGN channels	51
cbawgnc.m	Compute capacity for BAWGN channel	51
cawgnc.m	Compute capacity for AWGN channel	51
philog.m	Compute the $\log \phi$ function associated with the BAWGNC	51
phifun.m	Compute the ϕ function associated with the BAWGNC	51
gaussj2	Gaussian elimination over $GF(2)$	86
Hamsphere	Compute the number of points in a Hamming sphere	89
genstdarray.c	Generate a standard array for a code	91
progdetH15.m	Probability of error detection for $(15, 11)$ Hamming code	100
progdet.m	Probability of error detection for $(31, 21)$ Hamming code	100
polyadd.m	Add polynomials	116
polysub.m	Subtract polynomials	116
polymult.m	Multiply polynomials	116
polydiv.m	Divide polynomials (compute quotient and remainder)	116
polyaddm.m	Add polynomials modulo a number	116
polysubm.m	Subtract polynomials modulo a number	116
polymultm.m	Multiply polynomials modulo a number	116
primitive.txt	Table of primitive polynomials	155
BinLFSR.h	(lab, complete) Binary LFSR class	162
BinLFSR.cc	(lab, complete) Binary LFSR class	162
testBinLFSR.cc	(lab, complete) Binary LFSR class tester	162
MakeLFSR	(lab, complete) Makefile for tester	162
BinPolyDiv.h	(lab, complete) Binary polynomial division	162
BinPolyDiv.cc	(lab, incomplete) Binary polynomial division	162
testBinPolyDiv.cc	(lab, complete) Binary polynomial division test	162
gcd.c	A simple example of the Euclidean algorithm	181
crtgamma.m	Compute the gammas for the CRT	189

fromcrt.m	Convert back from CRT representation to an integer	189
tocrt.m	Compute the CRT representation of an integer	189
testcrt.m	An example of CRT calculations	189
testcrp.m	Test a polynomial CRT calculation	189
tocrtpoly.m	Compute the CRT representation of a polynomial	189
fromcrtpolym.	Compute a polynomial from a CRT representation	189
crtgammapoly.m	Compute the gammas for the CRT representation	189
primfind	Executable: Find primitive polynomials in $GF(p)[x]$	209
cyclomin	Executable: Cyclotomic cosets and minimal polynomials	217
ModAr.h	(lab, complete) Modulo arithmetic class	223
ModAr.cc	(lab, complete) Modulo arithmetic class	223
ModArnew.cc	Templatized Modulo arithmetic class	223
testmodarnew.cc	Templatized Modulo arithmetic class tester	223
testmodar1.cc	(lab, complete) Test Modulo arithmetic class	223
polynomialT.h	(lab, complete) Templatized polynomial class	223
polynomialT.cc	(lab, complete) Templatized polynomial class	223
testpoly1.cc	(lab, complete) Demonstrate templatized polynomial class	223
testgcdpoly.cc	(lab, complete) Test the polynomial GCD function	224
gcdpoly.cc	(lab, incomplete) Polynomial GCD function	224
GF2.h	(lab, complete) GF(2) class	224
GFNUM2m.h	(lab, complete) Galois field $GF(2^m)$ class	224
GFNUM2m.cc	(lab, incomplete) Galois field $GF(2^m)$ class	224
testgfnum.cc	(lab, complete) Test Galois field class	224
bchweight.m	Weight distribution of BCH code from weight of dual	240
bchdesigner	Executable: Design a t -error correcting binary BCH code	241
reedsolwt.m	Compute weight distribution for an (n, k) RS code	246
masseymodM.m	Return the shortest LFSR for data sequence	258
erase.mag	Erasur decoding example in magma	268
testBM.cc	(lab, complete) Test the Berlekamp-Massey algorithm	282
Chiensearch.h	(lab, complete) Chien Search class	283
Chiensearch.cc	(lab, incomplete) Chien Search class	283
testChien.cc	(lab, complete) Test the Chien Search class	283
BCHdec.h	(lab, complete) BCHdec decoder class	283
BCHdec.cc	(lab, incomplete) BCHdec decoder class	283
testBCH.cc	(lab, complete) BCHdec decoder class tester	283
RSenc.h	(lab, complete) RS encoder class header	284
RSenc.cc	(lab, complete) RS encoder class	284
RSdec.h	(lab, complete) RS decoder class header	284
RSdec.cc	(lab, incomplete) RS decoder class	284
testRS.cc	(lab, complete) Test RS decoder	284
rsencode.cc	(lab, complete) Encode a file of data using RS encoder	285
rsdecode.cc	(lab, complete) Decode a file of data using RS decoder	285
bsc.c	Executable: Simulate a binary symmetric channel	285
testpxy.cc	Demonstrate concepts relating to 2-variable polynomials	325
computekm.m	Compute K_m for the Guruswami-Sudan decoder	333
computeLm.cc	Compute the maximum list length for $GS(m)$ decoding	337
computeLm.m	Compute the maximum list length for $GS(m)$ decoding	337

testft.m	Test the Feng-Tzeng algorithm	341
fengtzeng.m	Poly. such that the first $l + 1$ columns are lin. dep.	341
invmodp.m	Compute the inverse of a number modulo p	341
testGS1.cc	Test the GS decoder (Kotter part)	346
kotter.cc	Kotter interpolation algorithm	346
testGS3.cc	Test the GS decoder	347
testGS5.cc	Test the GS decoder	350
kotter1.cc	Kotter algorithm with $m = 1$	350
testGS2.cc	Test the Roth-Ruckenstein algorithm	354
rothruck.cc	Roth-Ruckenstein algorithm (find y -roots)	354
rothruck.h	Roth-Ruckenstein algorithm (find y -roots)	354
Lbarex.m	Average performance of a $GS(m)$ decoder	357
computetm.m	Compute T_m , error correction capability for GS decoder	357
computeLbar.m	Avg. no. of codewords in sphere around random pt.	357
computeLm.m	Compute maximum length of list of $GF(m)$ decoder	357
pi2m1	Koetter-Vardy algorithm to map reliability to multiplicity	362
genrm.cc	Create a Reed-Muller generator matrix	376
rmdecex.m	$RM(1, 3)$ decoding example	381
hadex.m	Computation of H_8	382
testfht.cc	Test the fast Hadamard transform	383
fht.cc	Fast Hadamard transform	383
fht.m	Fast Hadamard transform	383
rmdecex2.m	$RM(2, 4)$ decoding example	387
testQR.cc	Example of arithmetic for QR code decoding	397
golaysimp.m	Derive equations for algebraic Golay decoder	401
testGolay.cc	Test the algebraic Golay decoder	401
golayrith.m	Arithmetic Golay decoder	402
plotbds.m	Plot bounds for binary codes	407
simplex1.m	Linear program solution to problems in standard form	413
pivottableau.m	Main function in simplex1.m	413
reducefree.m	Auxiliary linear programming function	413
restorefree.m	Auxiliary linear programming function	413
krawtchouk.m	Compute Krawtchouk polynomials recursively	415
lpboundex.m	Solve the linear programming for the LP bound	418
utiltkm.cc	Sort and random functions	440
utiltkm.h	Sort and random functions	440
concodequant.m	Compute the quantization of the Euclidean metric	486
chernoff1.m	Chernoff bounds for convolutional performance	502
plotconprob.m	Plot performance bounds for a convolutional code	504
finddfree	Executable: Find d_{free} for connection coefficients	506
teststack.m	Test the stack algorithm	515
stackalg.m	The stack algorithm for convolutional decoding	515
fanomet.m	Compute the Fano metric for convolutionally coded data	515
fanoalg.m	The Fano algorithm for convolutional decoding	517
BinConv.h	(lab, complete) Base class for binary convolutional encoder	526
BinConvFIR.h	(lab, complete) Binary feedforward convolutional encoder	526
BinConvFIR.cc	(lab, incomplete) Binary feedforward convolutional encoder	526

BinConvIIR.h	(lab, complete) Binary recursive convolutional encoder . . .	526
BinConvIIR.cc	(lab, incomplete) Binary recursive convolutional encoder . .	526
testconvenc	(lab, complete) Test convolutional encoders	526
Convdec.h	(lab, complete) Convolutional decoder class	528
Convdec.cc	(lab, incomplete) Convolutional decoder class	528
BinConvdec01.h	(lab, complete) Binary conv. decoder class, BSC metric . .	528
BinConvdec01.h	(lab, complete) Binary conv. decoder class, BSC metric . .	528
BinConvdecBPSK.h	(lab, complete) Bin. conv. decoder, soft metric with BPSK	528
BinConvdecBPSK.cc	(lab, complete) Bin. conv. decoder, soft metric with BPSK	528
testconvdec.cc	(lab, complete) Test the convolutional decoder	529
makeB.m	Make the B matrix for an example code	549
tcmt1.cc	Test the constellation for a rotationally invariant code . . .	549
tcrot2.cc	Test the constellation for a rotationally invariant code . . .	557
lattstuff.m	Generator matrices for $A_2, D_4, E_6, E_8, \Lambda_{16}, \Lambda_{24}$	563
voln.m	Compute volume of n -dimensional unit-radius sphere	563
latta2.m	Plot A_2 lattice	568
lattz2m	Plot Z_2 lattice	568
BCJR.h	(lab, complete) BCJR algorithm class header	629
BCJR.cc	(lab, incomplete) BCJR algorithm class	629
testbcjr.cc	(lab, complete) Test BCJR algorithm class	629
testtturbodec2.cc	(lab, complete) Test the turbo decoder	629
makgenfromA.m	Find systematic generator matrix for a parity check matrix .	635
gaussj2.m	Gauss-Jordan elimination over $GF(2)$ on a matrix	635
Agall.m	A parity check matrix for an LDPC code	637
Agall.txt	Sparse representation of the matrix	637
writesparse.m	Write a matrix into a file in sparse format	637
ldpc.m	Demonstrate LDPC decoding	648
galdecode.m	LDPC decoder (nonsparse representation)	648
ldpclogdec.m	Log-likelihood LDPC decoder	652
psifunc.m	Plot the Ψ function used in density evolution	656
densev1.m	An example of density evolution	658
densevtest.m	Plot density evolution results	658
Psi.m	Plot the Ψ function used in density evolution	658
Psiinv.m	Compute Ψ^{-1} used in density evolution	658
threshtab.m	Convert threshold table to E_b/N_0	658
ldpcsim.mat	LDPC decoder simulation results	660
exit1.m	Plot histograms of LDPC decoder outputs	660
loghist.m	Find histograms	660
exit3.m	Plot mutual information as a function of iteration	660
dotrajectory.m	Mutual information as a function of iteration	660
exit2.m	Plot EXIT chart	660
doexitchart.m	Take mutual information to EXIT chart	660
getinf.m	Convert data to mutual information	660
getinfs.m	Convert multiple data to mutual information	660
sparseHno4.m	Make a sparse check matrix with no cycles of girth 4	668
Asmall.txt	(lab, complete) A matrix in sparse representation	675
galdec.h	(lab, complete) LDPC decoder class header	675

galdec.cc	(lab, incomplete) LDPC decoder class	675
galtest.cc	(lab, complete) LDPC decoder class tester	675
ldpc.m	(lab, complete) Demonstrate LDPC decoding	675
galdecode.m	(lab, complete) LDPC decoder (not sparse representation) .	675
galtest2.cc	(lab, complete) Prob. of error plots for LDPC code	675
A1-2.txt	(lab, complete) Rate 1/2 parity check matrix, sparse format	675
A1-4.txt	(lab, complete) Rate 1/4 parity check matrix, sparse format	675
testgdl2.m	Test the generalized distributive law	695
gdl.m	A generalized distributive law function	695
fyx0.m	Compute $f(y x)$ for the GDL framework	695
fadeplot.m	Plot realizations of the amplitude of a fading channel . . .	712
jakes.m	Jakes method for computing amplitude of a fading channel	712
fadepbplot.m	Plot BPSK performance over Rayleigh fading channel . . .	714

List of Laboratory Exercises

Lab 1	Simulating a Communications Channel	53
Lab 2	Polynomial Division and Linear Feedback Shift Registers	161
Lab 3	CRC Encoding and Decoding	162
Lab 4	Programming the Euclidean Algorithm	223
Lab 5	Programming Galois Field Arithmetic	224
Lab 6	Programming the Berlekamp-Massey Algorithm	281
Lab 7	Programming the BCH Decoder	283
Lab 8	Reed-Solomon Encoding and Decoding	284
Lab 9	Programming Convolutional Encoders	526
Lab 10	Convolutional Decoders: The Viterbi Algorithm	528
Lab 11	Trellis-Coded Modulation Encoding and Decoding	578
Lab 12	Turbo Code Decoding	629
Lab 13	Programming an LDPC Decoder	674

List of Algorithms

1.1	Hamming Code Decoding	35
1.2	Outline for simulating a digital communications channel	53
1.3	Outline for simulating (n, k) -coded digital communications	53
1.4	Outline for simulating (n, k) Hamming-coded digital communications	54
4.1	Fast CRC encoding for a stream of bytes	152
4.2	Binary linear feedback shift register	162
4.3	Binary polynomial division	162
5.1	Extended Euclidean Algorithm	181
5.2	Modulo Arithmetic	223
5.3	Templatized Polynomials	223
5.4	Polynomial GCD	223
5.5	$GF(2^m)$ arithmetic	224
6.1	Massey's Algorithm (pseudocode)	258
6.2	Massey's Algorithm for Binary BCH Decoding	259
6.3	Test Berlekamp-Massey algorithm	282
6.4	Chien Search	283
6.5	BCH Decoder	283
6.6	Reed-Solomon Encoder Declaration	284
6.7	Reed-Solomon Decoder Declaration	284
6.8	Reed-Solomon Decoder Testing	284
6.9	Reed-Solomon File Encoder and Decoder	285
6.10	Binary Symmetric Channel Simulator	285
7.1	Welch-Berlekamp Interpolation	308
7.2	Welch-Berlekamp Interpolation, Modular Method	316
7.3	Welch-Berlekamp Interpolation, Modular Method v. 2	319
7.4	Welch-Berlekamp Interpolation, Modular Method v. 3	321
7.5	The Feng-Tzeng Algorithm	341
7.6	Kötters Interpolation for Guruswami-Sudan Decoder	346
7.7	Guruswami-Sudan Interpolation Decoder with $m = 1$ and $L = 1$	349
7.8	Roth-Ruckenstein Algorithm for Finding y -roots of $Q(x, y)$	353
7.9	Koetter-Vardy Algorithm for Mapping from Π to M	362
8.1	Decoding for $RM(1, m)$ Codes	381
8.2	Arithmetic Decoding of the Golay \mathcal{G}_{24} Code	402
11.1	Generalized Minimum Distance (GMD) Decoding	441
11.2	Chase-2 Decoder	445
11.3	Chase-3 Decoder	445
11.4	Ordered Statistic Decoding	449
12.1	The Viterbi Algorithm	475
12.2	The Stack Algorithm	515
12.3	Base Class for Binary Convolutional Encoder	526
12.4	Derived classes for FIR and IIR Encoders	526
12.5	Test program for convolutional encoders	526

12.6 Base Decoder Class Declarations	528
12.7 Convolutional decoder for binary (0,1) data	528
12.8 Convolutional decoder for BPSK data	528
12.9 Test the convolutional decoder	529
14.1 The BCJR (MAP) Decoding Algorithm, Probability Form	596
14.2 The Turbo Decoding Algorithm, Probability Form	601
14.3 BCJR algorithm	629
14.4 Test the turbo decoder algorithm	629
15.1 Iterative Decoding Algorithm for Binary LDPC Codes	648
15.2 Iterative Log Likelihood Decoding Algorithm for Binary LDPC Codes	652
15.3 LDPC decoder class declarations	675
15.4 Matlab code to test LDPC decoding	675
15.5 Make performance plots for LDPC codes	675

List of Figures

1.1	The binary entropy function $H_2(p)$.	4
1.2	A general framework for digital communications.	6
1.3	Signal constellation for BPSK.	10
1.4	Juxtaposition of signal waveforms.	11
1.5	Two-dimensional signal space.	12
1.6	8-PSK signal constellation.	13
1.7	Correlation processing (equivalent to matched filtering).	15
1.8	Conditional densities in BPSK modulation.	18
1.9	Distributions when two signals are sent in Gaussian noise.	20
1.10	Probability of error for BPSK signaling.	21
1.11	Probability of error bound for 8-PSK modulation.	22
1.12	A binary symmetric channel.	23
1.13	Communication system diagram and BSC equivalent.	25
1.14	Energy for a coded signal.	26
1.15	Probability of error for coded bits, before correction.	27
1.16	A (3, 1) binary repetition code.	29
1.17	A representation of decoding spheres.	30
1.18	Performance of the (3, 1) and (11, 1) repetition code over BSC.	32
1.19	Performance of the (7, 4) Hamming code in the AWGN channel.	37
1.20	The trellis of a (7, 4) Hamming code.	39
1.21	The Tanner graph for a (7, 4) Hamming code.	39
1.22	Capacities of AWGNC, BAWGNC, and BSC.	46
1.23	Relationship between input and output entropies for a channel.	48
1.24	Capacity lower bounds on P_b as a function of SNR.	52
1.25	Regions for bounding the Q function.	57
2.1	An illustration of cosets.	67
2.2	A lattice partitioned into cosets.	72
3.1	Error detection performance for a (15,11) Hamming code.	100
3.2	Demonstrating modifications on a Hamming code.	107
4.1	A circuit for multiplying two polynomials, last-element first.	128
4.2	High-speed circuit for multiplying two polynomials, last-element first.	129
4.3	A circuit for multiplying two polynomials, first-element first.	129
4.4	High-speed circuit for multiplying two polynomials, first-element first.	130
4.5	A circuit to perform polynomial division.	131
4.6	A circuit to divide by $g(x) = x^5 + x + 1$.	131
4.7	Realizing $h(x)/g(x)$ (first-element first), controller canonical form.	133
4.8	Realizing $h(x)/g(x)$ (first-element first), observability form.	134
4.9	Realization of $H(x) = (1 + x)/(1 + x^3 + x^4)$, controller form.	134
4.10	Realization of $H(x) = (1 + x)/(1 + x^3 + x^4)$, observability form.	134

4.11	Nonsystematic encoding of cyclic codes.	135
4.12	Circuit for systematic encoding using $g(x)$	136
4.13	Systematic encoder for $(7, 4)$ code with generator $g(x) = 1 + x + x^3$	136
4.14	A systematic encoder using the parity check polynomial.	137
4.15	A systematic encoder for the Hamming code using $h(x)$	137
4.16	A syndrome computation circuit for a cyclic code example.	139
4.17	Cyclic decoder with $r(x)$ shifted in the left end of syndrome register.	140
4.18	Decoder for a $(7,4)$ Hamming code, input on the left.	141
4.19	Cyclic decoder when $r(x)$ is shifted in right end of syndrome register.	143
4.20	Hamming decoder with input fed into right end of the syndrome register.	144
4.21	Meggitt decoders for the $(31,26)$ Hamming code.	145
4.22	Multiply $r(x)$ by $\rho(x)$ and compute the remainder modulo $g(x)$	146
4.23	Decoder for a shortened Hamming code.	147
4.24	Linear feedback shift register.	154
4.25	Linear feedback shift register with $g(x) = 1 + x + x^2 + x^4$	155
4.26	Linear feedback shift register with $g(x) = 1 + x + x^4$	156
4.27	Linear feedback shift register, reciprocal polynomial convention.	158
4.28	Another LFSR circuit.	169
4.29	An LFSR with state labels.	169
5.1	LFSR labeled with powers of α to illustrate Galois field elements.	200
5.2	Multiplication of β by α	205
5.3	Multiplication of an arbitrary β by α^4	205
5.4	Multiplication of β by an arbitrary field element.	206
5.5	Subfield structure of $GF(2^{24})$	207
6.1	Chien search algorithm.	249
7.1	Remainder computation when errors are in message locations.	295
7.2	Comparing BD, ML, and list decoding.	323
7.3	K_m as a function of m for a $(32, 8)$ Reed-Solomon code.	333
7.4	Fraction of errors corrected as a function of rate.	335
7.5	An example of the Roth-Ruckenstein Algorithm over $GF(5)$	355
7.6	An example of the Roth-Ruckenstein Algorithm over $GF(5)$ (cont'd).	356
7.7	Convergence of \bar{M} to Π	363
7.8	Computing the reliability function.	364
8.1	An encoder circuit for a $RM(1, 3)$ code.	379
8.2	Signal flow diagram for the fast Hadamard transform.	384
8.3	Binary adjacency relationships in three and four dimensions.	388
8.4	Planes shaded to represent the equations orthogonal on bit m_{34}	388
8.5	Parity check geometric descriptions for vectors of the $RM(2, 4)$ code.	390
8.6	Parity check geometric descriptions for vectors of the $RM(2, 4)$ code.	391
9.1	Comparison of lower bound and various upper bounds.	407
9.2	A linear programming problem.	413
9.3	Finding Stirling's formula.	422

10.1	A 3×4 interleaver and deinterleaver.	426
10.2	A cross interleaver and deinterleaver system.	428
10.3	The CD recording and data formatting process.	429
10.4	The error correction encoding in the compact disc standard.	429
10.5	The product code $C_1 \times C_2$	431
10.6	A concatenated code.	432
10.7	Deep-space concatenated coding system.	433
10.8	Error trapping decoder for burst-error correcting codes.	436
11.1	Signal labels for soft-decision decoding.	440
12.1	A rate $R = 1/2$ convolutional encoder.	453
12.2	A systematic $R = 1/2$ encoder.	454
12.3	A systematic $R = 2/3$ encoder.	455
12.4	A systematic $R = 2/3$ encoder with more efficient hardware.	455
12.5	Encoder, state diagram, and trellis for $G(x) = [1 + x^2, 1 + x + x^2]$	458
12.6	State diagram and trellis for a rate $R = 2/3$ systematic encoder.	459
12.7	A feedforward $R = 2/3$ encoder.	460
12.8	A less efficient feedforward $R = 2/3$ encoder.	461
12.9	Processing stages for a convolutional code.	469
12.10	Notation associated with a state transition.	472
12.11	The Viterbi step: Select the path with the best metric.	474
12.12	Path through trellis corresponding to true sequence.	476
12.13	Add-compare-select Operation.	481
12.14	A two-bit quantization of the soft-decision metric.	485
12.15	Quantization thresholds for 4- and 8-level quantization.	487
12.16	Bit error rate for different constraint lengths.	488
12.17	Bit error rate for various quantization and window lengths.	489
12.18	Viterbi algorithm performance as a function of quantizer threshold spacing.	490
12.19	BER performance as a function of truncation block length.	490
12.20	Error events due to merging paths.	491
12.21	Two decoding examples.	492
12.22	The state diagram and graph for diverging/remerging paths.	494
12.23	Rules for simplification of flow graphs.	495
12.24	Steps simplifying the flow graph for a convolutional code.	495
12.25	State diagram labeled with input/output weight and branch length.	496
12.26	A state diagram to be enumerated.	497
12.27	Performance of a $(3, 1)$ convolutional code with $d_{\text{free}} = 5$	505
12.28	Trellises for a punctured code.	509
12.29	A tree representation for a rate $R = 1/2$ code.	512
12.30	Stack contents for stack algorithm decoding example.	516
12.31	Flowchart for the Fano algorithm.	519
12.32	The trellis of a $(7, 4)$ Hamming code.	524
12.33	A systematic encoder for a $(7, 4, 3)$ Hamming code.	524
12.34	A trellis for a cyclically encoded $(7,4,3)$ Hamming code.	525
12.35	State diagram and trellis	527
13.1	PSK signal constellations.	537

13.2	QAM Signal constellations (overlaid).	537
13.3	Three communication scenarios.	538
13.4	Set partitioning of an 8-PSK signal.	539
13.5	$R = 2/3$ trellis coded modulation example.	540
13.6	A TCM encoder employing subset selection and a four-state trellis.	542
13.7	An 8-state trellis for 8-PSK TCM.	543
13.8	Block diagram of a TCM encoder.	544
13.9	Set partitioning on a 16-QAM constellation.	545
13.10	Partition for 8-ASK signaling.	546
13.11	A correct path and an error path.	548
13.12	Example trellis for four-state code.	549
13.13	Trellis coder circuit.	552
13.14	TCM encoder for QAM constellations.	553
13.15	Mapping of edge (i, j) to edge $(f_\phi(i), f_\phi(j))$.	556
13.16	Encoder circuit for rotationally invariant TCM code.	557
13.17	Trellis for the rotationally invariant code of Figure 13.16	558
13.18	32-cross constellation for rotationally invariant TCM code.	558
13.19	A portion of the lattice \mathbb{Z}^2 and its cosets.	564
13.20	Hexagonal lattice.	565
13.21	\mathbb{Z}^2 and its partition chain and cosets.	568
13.22	Block diagram for a trellis lattice coder.	569
13.23	Lattice and circular boundaries for various constellations.	570
13.24	16-state trellis encoder for use with V.34 standard.	572
13.25	Trellis diagram of V.34 encoder.	573
13.26	The 192-point constellation employed in the V.34 standard.	575
13.27	Partition steps for the V.34 signal constellation.	576
13.28	Orbits of some of the points under rotation.	576
14.1	Decoding results for a $(37,21,65536)$ code.	583
14.2	Block diagram of a turbo encoder.	585
14.3	Block diagram of a turbo encoder with puncturing.	586
14.4	Example turbo encoder with $G(x) = 1/1 + x^2$.	587
14.5	Block diagram of a turbo decoder.	587
14.6	Processing stages for BCJR algorithm.	589
14.7	One transition of the trellis for the encoder.	590
14.8	A log likelihood turbo decoder.	604
14.9	Trellis with metric differences and bits for SOVA.	612
14.10	State sequences for an encoding example.	616
14.11	Arrangements of $n_1 = 3$ detours in a sequence of length $N = 7$.	616
14.12	A 6×6 "square" sequence written into the 120×120 interleaver.	618
14.13	Variables used in iterative decoder for EXIT chart analysis.	620
14.14	Qualitative form of the transfer characteristic $I_E = T(I_A)$.	622
14.15	Trajectories of mutual information in iterated decoding.	623
14.16	Turbo BCH encoding.	624
14.17	Structure of an implementation of a parallel concatenated code.	624
14.18	A trellis for a cyclically encoded $(7,4,3)$ Hamming code.	625
14.19	Framework for a turbo equalizer.	627