This Page Intentionally Left Blank
Error Correction Coding
This Page Intentionally Left Blank
Error Correction Coding
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 dimensionally 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, 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


Other aspects of the book include the following:

Matlab 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 it 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 □.

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 xi
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
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Trellis Representation</td>
<td>38</td>
</tr>
<tr>
<td>The Tanner Graph Representation</td>
<td>38</td>
</tr>
<tr>
<td>1.10 The Basic Questions</td>
<td>39</td>
</tr>
<tr>
<td>1.11 Historical Milestones of Coding Theory</td>
<td>40</td>
</tr>
<tr>
<td>1.12 A Bit of Information Theory</td>
<td>40</td>
</tr>
<tr>
<td>1.12.1 Definitions for Discrete Random Variables</td>
<td>40</td>
</tr>
<tr>
<td>Entropy and Conditional Entropy</td>
<td>40</td>
</tr>
<tr>
<td>Relative Entropy, Mutual Information, and Channel Capacity</td>
<td>41</td>
</tr>
<tr>
<td>1.12.2 Definitions for Continuous Random Variables</td>
<td>43</td>
</tr>
<tr>
<td>1.12.3 The Channel Coding Theorem</td>
<td>45</td>
</tr>
<tr>
<td>1.12.4 “Proof” of the Channel Coding Theorem</td>
<td>45</td>
</tr>
<tr>
<td>1.12.5 Capacity for the Continuous-Time AWGN Channel</td>
<td>49</td>
</tr>
<tr>
<td>1.12.6 Transmission at Capacity with Errors</td>
<td>51</td>
</tr>
<tr>
<td>1.12.7 The Implication of the Channel Coding Theorem</td>
<td>52</td>
</tr>
<tr>
<td>Lab 1 Simulating a Communications Channel</td>
<td>53</td>
</tr>
<tr>
<td>Objective</td>
<td>53</td>
</tr>
<tr>
<td>Background</td>
<td>53</td>
</tr>
<tr>
<td>Use of Coding in Conjunction with the BSC</td>
<td>53</td>
</tr>
<tr>
<td>Assignment</td>
<td>54</td>
</tr>
<tr>
<td>Programming Part</td>
<td>54</td>
</tr>
<tr>
<td>Resources and Implementation Suggestions</td>
<td>54</td>
</tr>
<tr>
<td>1.13 Exercises</td>
<td>56</td>
</tr>
<tr>
<td>1.14 References</td>
<td>60</td>
</tr>
</tbody>
</table>

### Part II Block Codes

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 Groups and Vector Spaces</td>
<td>62</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>62</td>
</tr>
<tr>
<td>2.2 Groups</td>
<td>62</td>
</tr>
<tr>
<td>2.2.1 Subgroups</td>
<td>65</td>
</tr>
<tr>
<td>2.2.2 Cyclic Groups and the Order of an Element</td>
<td>66</td>
</tr>
<tr>
<td>2.2.3 Cosets</td>
<td>67</td>
</tr>
<tr>
<td>2.2.4 Lagrange’s Theorem</td>
<td>68</td>
</tr>
<tr>
<td>2.2.5 Induced Operations; Isomorphism</td>
<td>69</td>
</tr>
<tr>
<td>2.2.6 Homomorphism</td>
<td>72</td>
</tr>
<tr>
<td>2.3 Fields: A Prelude</td>
<td>73</td>
</tr>
<tr>
<td>2.4 Review of Linear Algebra</td>
<td>75</td>
</tr>
<tr>
<td>2.5 Exercises</td>
<td>80</td>
</tr>
<tr>
<td>2.6 References</td>
<td>82</td>
</tr>
<tr>
<td>3 Linear Block Codes</td>
<td>83</td>
</tr>
<tr>
<td>3.1 Basic Definitions</td>
<td>83</td>
</tr>
<tr>
<td>3.2 The Generator Matrix Description of Linear Block Codes</td>
<td>84</td>
</tr>
<tr>
<td>3.2.1 Rudimentary Implementation</td>
<td>86</td>
</tr>
<tr>
<td>3.3 The Parity Check Matrix and Dual Codes</td>
<td>86</td>
</tr>
<tr>
<td>3.3.1 Some Simple Bounds on Block Codes</td>
<td>88</td>
</tr>
<tr>
<td>3.4 Error Detection and Correction over Hard-Input Channels</td>
<td>90</td>
</tr>
</tbody>
</table>
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 Erasure 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
    4.9.3.1 Last-Element-First Processing ......................... 128
    4.9.3.2 First-Element-First Processing ......................... 128
    4.9.4 Polynomial division ....................................... 129
    4.9.4.1 Last-Element-First Processing ......................... 129
    4.9.5 Simultaneous Polynomial Division and Multiplication .... 132
    4.9.5.1 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

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

Objective ................................................................. 162
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

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 $\phi$ Function ............................................. 185
  5.2.6 Some Cryptographic Payoff  
      Fermat's Little Theorem ........................................ 186
      RSA Encryption .................................................. 187
  5.3 The Chinese Remainder Theorem ................................ 188
    5.3.1 The CRT and Interpolation  
    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


6 BCH and Reed-Solomon Codes: Designer Cyclic Codes

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
CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preliminary Exercises</td>
<td>281</td>
</tr>
<tr>
<td>Programming Part</td>
<td>281</td>
</tr>
<tr>
<td>Resources and Implementation Suggestions</td>
<td>282</td>
</tr>
<tr>
<td><strong>Lab 7 Programming the BCH Decoder</strong></td>
<td>283</td>
</tr>
<tr>
<td>Objective</td>
<td>283</td>
</tr>
<tr>
<td>Preliminary Exercises</td>
<td>283</td>
</tr>
<tr>
<td>Programming Part</td>
<td>283</td>
</tr>
<tr>
<td>Resources and Implementation Suggestions</td>
<td>283</td>
</tr>
<tr>
<td>Follow-On Ideas and Problems</td>
<td>284</td>
</tr>
<tr>
<td><strong>Lab 8 Reed-Solomon Encoding and Decoding</strong></td>
<td>284</td>
</tr>
<tr>
<td>Objective</td>
<td>284</td>
</tr>
<tr>
<td>Background</td>
<td>284</td>
</tr>
<tr>
<td>Programming Part</td>
<td>284</td>
</tr>
<tr>
<td>Appendix 6.A Proof of Newton's Identities</td>
<td>285</td>
</tr>
<tr>
<td>6.10 Exercises</td>
<td>287</td>
</tr>
<tr>
<td>6.11 References</td>
<td>291</td>
</tr>
</tbody>
</table>

7 Alternate Decoding Algorithms for Reed-Solomon Codes 293

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1 Introduction: Workload for Reed-Solomon Decoding</td>
<td>293</td>
</tr>
<tr>
<td>7.2 Derivations of Welch-Berlekamp Key Equation</td>
<td>293</td>
</tr>
<tr>
<td>7.2.1 The Welch-Berlekamp Derivation of the WB Key Equation</td>
<td>294</td>
</tr>
<tr>
<td>7.2.2 Derivation From the Conventional Key Equation</td>
<td>298</td>
</tr>
<tr>
<td>7.3 Finding the Error Values</td>
<td>300</td>
</tr>
<tr>
<td>7.4 Methods of Solving the WB Key Equation</td>
<td>302</td>
</tr>
<tr>
<td>7.4.1 Background: Modules</td>
<td>302</td>
</tr>
<tr>
<td>7.4.2 The Welch-Berlekamp Algorithm</td>
<td>303</td>
</tr>
<tr>
<td>7.4.3 Modular Solution of the WB Key Equation</td>
<td>310</td>
</tr>
<tr>
<td>7.5 Erasure Decoding with the Welch-Berlekamp Key Equation</td>
<td>321</td>
</tr>
<tr>
<td>7.6 The Guruswami-Sudan Decoding Algorithm and Soft RS Decoding</td>
<td>322</td>
</tr>
<tr>
<td>7.6.1 Bounded Distance, ML, and List Decoding</td>
<td>322</td>
</tr>
<tr>
<td>7.6.2 Error Correction by Interpolation</td>
<td>323</td>
</tr>
<tr>
<td>7.6.3 Polynomials in Two Variables</td>
<td>324</td>
</tr>
<tr>
<td>Degree and Monomial Order</td>
<td>325</td>
</tr>
<tr>
<td>Zeros and Multiple Zeros</td>
<td>328</td>
</tr>
<tr>
<td>7.6.4 The GS Decoder: The Main Theorems</td>
<td>330</td>
</tr>
<tr>
<td>The Interpolation Theorem</td>
<td>331</td>
</tr>
<tr>
<td>The Factorization Theorem</td>
<td>331</td>
</tr>
<tr>
<td>The Correction Distance</td>
<td>333</td>
</tr>
<tr>
<td>The Number of Polynomials in the Decoding List</td>
<td>335</td>
</tr>
<tr>
<td>7.6.5 Algorithms for Computing the Interpolation Step</td>
<td>337</td>
</tr>
<tr>
<td>Finding Linearly Dependent Columns: The Feng-Tzeng Algorithm</td>
<td>338</td>
</tr>
<tr>
<td>Finding the Intersection of Kernels: The Kötter Algorithm</td>
<td>342</td>
</tr>
<tr>
<td>7.6.6 A Special Case: $m = 1$ and $L = 1$</td>
<td>348</td>
</tr>
<tr>
<td>7.6.7 The Roth-Ruckenstein Algorithm</td>
<td>350</td>
</tr>
<tr>
<td>What to Do with Lists of Factors?</td>
<td>354</td>
</tr>
<tr>
<td>7.6.8 Soft-Decision Decoding of Reed-Solomon Codes</td>
<td>358</td>
</tr>
<tr>
<td>Notation</td>
<td>358</td>
</tr>
</tbody>
</table>
## CONTENTS

- 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 ≥ 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 423 Golay Code ................ 400
      Arithmetic Decoding of the 424 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

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

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

## 12 Convolutional Codes

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
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background</td>
<td>674</td>
</tr>
<tr>
<td>Assignment</td>
<td>675</td>
</tr>
<tr>
<td>Numerical Considerations</td>
<td>675</td>
</tr>
<tr>
<td>15.15 Exercises</td>
<td>676</td>
</tr>
<tr>
<td>15.16 References</td>
<td>679</td>
</tr>
<tr>
<td><strong>16 Decoding Algorithms on Graphs</strong></td>
<td>680</td>
</tr>
<tr>
<td>16.1 Introduction</td>
<td>680</td>
</tr>
<tr>
<td>16.2 Operations in Semirings</td>
<td>681</td>
</tr>
<tr>
<td>16.3 Functions on Local Domains</td>
<td>681</td>
</tr>
<tr>
<td>16.4 Factor Graphs and Marginalization</td>
<td>686</td>
</tr>
<tr>
<td>16.4.1 Marginalizing on a Single Variable</td>
<td>687</td>
</tr>
<tr>
<td>16.4.2 Marginalizing on All Individual Variables</td>
<td>691</td>
</tr>
<tr>
<td>16.5 Applications to Coding</td>
<td>694</td>
</tr>
<tr>
<td>16.5.1 Block Codes</td>
<td>694</td>
</tr>
<tr>
<td>16.5.2 Modifications to Message Passing for Binary Variables</td>
<td>695</td>
</tr>
<tr>
<td>16.5.3 Trellis Processing and the Forward/Backward Algorithm</td>
<td>696</td>
</tr>
<tr>
<td>16.5.4 Turbo Codes</td>
<td>699</td>
</tr>
<tr>
<td>16.6 Summary of Decoding Algorithms on Graphs</td>
<td>699</td>
</tr>
<tr>
<td>16.7 Transformations of Factor Graphs</td>
<td>700</td>
</tr>
<tr>
<td>16.7.1 Clustering</td>
<td>700</td>
</tr>
<tr>
<td>16.7.2 Stretching Variable Nodes</td>
<td>701</td>
</tr>
<tr>
<td>16.7.3 Exact Computation of Graphs with Cycles</td>
<td>702</td>
</tr>
<tr>
<td>16.8 Exercises</td>
<td>706</td>
</tr>
<tr>
<td>16.9 References</td>
<td>708</td>
</tr>
<tr>
<td><strong>Part V Space-Time Coding</strong></td>
<td>709</td>
</tr>
<tr>
<td><strong>17 Fading Channels and Space-Time Codes</strong></td>
<td>710</td>
</tr>
<tr>
<td>17.1 Introduction</td>
<td>710</td>
</tr>
<tr>
<td>17.2 Fading Channels</td>
<td>710</td>
</tr>
<tr>
<td>17.2.1 Rayleigh Fading</td>
<td>712</td>
</tr>
<tr>
<td>17.3 Diversity Transmission and Reception: The MIMO Channel</td>
<td>714</td>
</tr>
<tr>
<td>17.3.1 The Narrowband MIMO Channel</td>
<td>716</td>
</tr>
<tr>
<td>17.3.2 Diversity Performance with Maximal-Ratio Combining</td>
<td>717</td>
</tr>
<tr>
<td>17.4 Space-Time Block Codes</td>
<td>719</td>
</tr>
<tr>
<td>17.4.1 The Alamouti Code</td>
<td>719</td>
</tr>
<tr>
<td>17.4.2 A More General Formulation</td>
<td>721</td>
</tr>
<tr>
<td>17.4.3 Performance Calculation</td>
<td>721</td>
</tr>
<tr>
<td>17.4.3 Real Orthogonal Designs</td>
<td>723</td>
</tr>
<tr>
<td>17.4.3 Encoding and Decoding Based on Orthogonal Designs</td>
<td>724</td>
</tr>
<tr>
<td>17.4.3 Generalized Real Orthogonal Designs</td>
<td>726</td>
</tr>
<tr>
<td>17.4.4 Complex Orthogonal Designs</td>
<td>727</td>
</tr>
<tr>
<td>17.5 Future Work</td>
<td>728</td>
</tr>
<tr>
<td>17.6 How Many Antennas?</td>
<td>732</td>
</tr>
</tbody>
</table>
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
repcodeprob.m Compute error probability for (n, 1) repetition codes .... 32
testrepcode.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 BAWGNC channel ............ 45
h2.m Compute the binary entropy function ............................ 45
plotcbawn2.m Plot capacity for AWGN and BAWGNC channels .......... 51
cbawgnc.m Compute capacity for BAWGNC 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 \( \phi \) function associated with the BAWGNC ... 51
gaussj2 Gaussian elimination over \( GF(2) \) .......................... 86
Hammsphere 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
crtgammamapoly.m  Compute the gammas for the CRT representation . 189
primfind
cyclomin
ModAr.h
ModAr.cc
ModArnew.cc
 testmodaronew.cc
 testmodarl.cc
 polynomialT.h
 polynomialT.cc
testpoly1.cc
testgdpoly.cc
gcdpoly.cc
GP2.h
GFNUM2m.h
GFNUM2m.cc
testgfnum.cc
bchweight.m
bchdesigner
reedsolwt.m
masseymodM.m
erase.mag
testBM.cc
Chiensearch.h
Chiensearch.cc
testChien.cc
BCHdec.h
BCHdec.cc
testBCH.cc
RSenc.h
RSenc.cc
RSdec.h
RSdec.cc
testRS.cc
rsencode.cc
rsdecode.cc
bs.c
testpxy.cc
computekm.m
computeLm.cc
computeLm.m

Executable: Find primitive polynomials in $GF(p)[x]$ . . . . 209

Executable: Cyclotomic cosets and minimal polynomials . . . . 217
(lab, complete) Modulo arithmetic class . . . . . . . . . . . . . . . 223
(lab, complete) Modulo arithmetic class . . . . . . . . . . . . . . . 223
Templatized Modulo arithmetic class . . . . . . . . . . . . . . . . 223
Templatized Modulo arithmetic class tester . . . . . . . . . . . . . 223
(lab, complete) Test Modulo arithmetic class . . . . . . . . . . . . . 223
(lab, complete) Test the polynomial GCD function . . . . . . . . 224
(lab, incomplete) Polynomial GCD function . . . . . . . . . . . . . 224
(lab, complete) GF(2) class . . . . . . . . . . . . . . . . . . . . . . 224
(lab, complete) Galois field $GF(2^m)$ class . . . . . . . . . . . . . 224
(lab, incomplete) Galois field $GF(2^m)$ class . . . . . . . . . . . . 224
(lab, complete) Test Galois field class . . . . . . . . . . . . . . . . 224
Weight distribution of BCH code from weight of dual . . . . . . 240

Executable: Design a $t$-error correcting binary BCH code . . 241
Compute weight distribution for an $(n, k)$ RS code . . . . . . 246
Return the shortest LFSR for data sequence . . . . . . . . . . . . 258
Erasure decoding example in magma . . . . . . . . . . . . . . . . 268
(lab, complete) Test the Berlekamp-Massey algorithm . . . . . . . . 282
(lab, complete) Chien Search class . . . . . . . . . . . . . . . . . 283
(lab, incomplete) Chien Search class . . . . . . . . . . . . . . . . 283
(lab, complete) Test the Chien Search class . . . . . . . . . . . . . 283
(lab, complete) BCHdec decoder class . . . . . . . . . . . . . . . . 283
(lab, incomplete) BCHdec decoder class . . . . . . . . . . . . . . . 283
(lab, complete) BCHdec decoder class tester . . . . . . . . . . . . . 283
(lab, complete) RS encoder class header . . . . . . . . . . . . . . . 284
(lab, complete) RS encoder class . . . . . . . . . . . . . . . . . . . 284
(lab, complete) RS decoder class header . . . . . . . . . . . . . . . 284
(lab, complete) RS decoder class . . . . . . . . . . . . . . . . . . . 284
(lab, complete) Test RS decoder . . . . . . . . . . . . . . . . . . . 284
(lab, complete) Encode a file of data using RS encoder . . . . . . 285
(lab, complete) Decode a file of data using RS decoder . . . . . . 285

Executable: Simulate a binary symmetric channel . . . . . . . . 285
Demonstrate concepts relating to 2-variable polynomials . . . . 325
Compute $K_m$ for the Guruswami-Sudan decoder . . . . . . . . . 333
Compute the maximum list length for $GS(m)$ decoding . . . . . 337
Compute the maximum list length for $GS(m)$ decoding . . . . . 337