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

WILEY-INTERSCIENCE
A JOHN WILEY & SONS, INC., PUBLICATION

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

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.

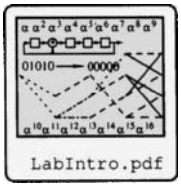
Some of the major pedagogical features of the book are as follows.

- 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.

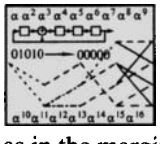
These labs also help with the coverage of the course material. In my course I am able to offload classroom instruction of some topics for students to read, with the assurance that the students will learn it solidly on their own as they implement it. (The Euclidean algorithm is one of these topics in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

[1] Matlab is a registered trademard 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.

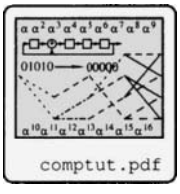
Chapter 14: Basic definition and the BCJR algorithm.

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 □.



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

This Page Intentionally Left Blank

List of Program Files

List of Laboratory Exercises

List of Algorithms

List of Figures