On the Construction of Some Capacity-Approaching Coding Schemes

by

Sae-Young Chung

Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2000

© Massachusetts Institute of Technology 2000. All rights reserved.

Author .....
Department of Electrical Engineering and Computer Science
September 1, 2000

Certified by .....
G. David Forney, Jr.
Adjunct Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

On the Construction of Some Capacity-Approaching Coding Schemes

by

Sae-Young Chung

Submitted to the Department of Electrical Engineering and Computer Science
on September 1, 2000, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

This thesis proposes two constructive methods of approaching the Shannon limit very closely. Interestingly, these two methods operate in opposite regions, one has a block length of one and the other has a block length approaching infinity.

The first approach is based on novel memoryless joint source-channel coding schemes. We first show some examples of sources and channels where no coding is optimal for all values of the signal-to-noise ratio (SNR). When the source bandwidth is greater than the channel bandwidth, joint coding schemes based on space-filling curves and other families of curves are proposed. For uniform sources and modulo channels, our coding scheme based on space-filling curves operates within 1.1 dB of Shannon's rate-distortion bound. For Gaussian sources and additive white Gaussian noise (AWGN) channels, we can achieve within 0.9 dB of the rate-distortion bound.

The second scheme is based on low-density parity-check (LDPC) codes. We first demonstrate that we can translate threshold values of an LDPC code between channels accurately using a simple mapping. We develop some models for density evolution from this observation, namely erasure-channel, Gaussian-capacity, and reciprocal-channel approximations. The reciprocal-channel approximation, based on dualizing LDPC codes, provides a very accurate model of density evolution for the AWGN channel. We also develop another approximation method, Gaussian approximation, which enables us to visualize infinite-dimensional density evolution and optimization of LDPC codes. We also develop other tools to better understand density evolution. Using these tools, we design some LDPC codes that approach the Shannon limit extremely closely. For multilevel AWGN channels, we design a rate 1/2 code that has a threshold within 0.0063 dB of the Shannon limit of the noisiest level. For binary-input AWGN channels, our best rate 1/2 LDPC code has a threshold within 0.0045 dB of the Shannon limit. Simulation results show that we can achieve within 0.04 dB of the Shannon limit at a bit error rate of 10^{-6} using a block length of 10^7.

Thesis Supervisor: G. David Forney, Jr.
Title: Adjunct Professor of Electrical Engineering and Computer Science

To Mom, Dad, Sylvia, and Christopher

Acknowledgments

I would like to express my sincerest gratitude to my supervisor Professor David Forney, for his guidance and insights throughout this thesis and my education at MIT. His engineering intuition and experience inspired my interest in channel coding and guided my research. I am extremely grateful to him for all his ultra-prompt help and for his numerous proofreads that greatly improved this work. This thesis would not have been possible without his guidance.

I wish to thank Professor Mitch Trott who was my supervisor during my first two years at MIT. His enthusiasm for research enabled me to work on the joint source and channel coding part of this thesis. I deeply appreciate his insightful ideas and his many hours spent with me discussing my research.

I would also like to thank my thesis committee; Professor Sanjoy Mitter and Dr. Tom Richardson for their insightful suggestions, which improved this thesis in many ways.

I was very lucky to work with Dr. Tom Richardson and Professor Rüdiger Urbanke, whose insightful ideas and landmark work on low-density parity-check codes motivated and inspired me in many ways. I am extremely grateful to them for all their support and many useful discussions for the last two years including my internship at Bell Labs during the Summer of 1999.

I also wish to thank Professor Robert Gallager, my academic advisor. His excellent advice and counselling helped me adapt to the research environment at LIDS.

I wish to thank my labmates at LIDS for their help and friendship, including Ibrahim Abou-Faycal, Louay Bazzi, Randy Berry, Choongyeun Cho, Jihwan Choi, Aaron Cohen, Hisham Kassab, Andrew Kim, Junmo Kim, Thierry Klein, Anant Sahai, Brett Schein, Sekhar Tatikonda, Dewey Tucker, Sean Warnick, Edmund Yeh, and Won Yoon. LIDS has been a great environment thanks to such a group of highly talented people.

Finally, I would like to express special thanks to my family. I thank my parents for their love, help and encouragement which made it possible for me to get a doctoral degree. No words are enough to show my appreciation for them. My wife Sylvia has always supported me during my five years at MIT. Her true love and extreme patience have made this thesis possible. I am especially grateful to her for taking such a good care of our newborn son Christopher. I also wish to thank Sylvia's parents and many of her uncles who helped us in many ways. I greatly appreciate their thoughtful support.

This research was partially supported by NSF grant NCR-9314341 and by Lucent Technologies.

Contents

1 Introduction **25**

1.1 Motivation 25

1.2 Joint Source and Channel Coding 27

1.3 Low-Density Parity-Check Codes 29

1.4 Thesis Organization 32

2 Matched Sources and Channels **35**

2.1 Source and Channel Model 35

2.2 Shannon Limit 37

2.2.1 Optimal Decoder 38

2.2.2 Optimal Encoder 38

2.3 Matched Sources and Channels 41

2.3.1 Examples of Matched Sources and Channels 46

2.4 A Uniform Source and Its Rate-Distortion Function 48

2.5 A Modulo-Additive Noise Channel and Its Capacity 53

2.6 Optimal Joint Coding for the Uniform Source and the MAWN Channel 55

2.7 Summary 56

3 Joint Source-Channel Coding Using Space-Filling Curves **57**

3.1 Introduction 57

3.1.1 Bandwidth Expansion Using Curves 57

3.1.2 Bandwidth Reduction Using Curves 58

3.1.3 Bandwidth Reduction Using Numerically Optimized Vector Quantizers . . . 60
3.2 Space-Filling Curves . . . 67
3.3 Joint Coding for Uniform Sources . . . 74
3.3.1 Joint Coding Using Lattice Vector Quantizers . . . 75
3.3.2 Joint Coding Using Approximate Curves . . . 82
3.4 Joint Coding for Gaussian Sources . . . 86
3.4.1 Joint Coding Using Stretched Space-Filling Curves . . . 87
3.4.2 Joint Coding Using Spiral Curves . . . 89
3.5 Summary . . . 96

4 Normalization of Channels **97**
4.1 Channel Model . . . 97
4.2 Normalization of Symmetric Channels . . . 100
4.3 Summary . . . 106

5 Low-Density Parity-Check Codes and Decoding **107**
5.1 Codes on Graphs and Normal Realization . . . 107
5.2 The Sum-Product Algorithm and Density Evolution . . . 110
5.2.1 The Sum-Product Algorithm . . . 110
5.2.2 Density Evolution . . . 111
5.2.3 Iterative Normalization . . . 117
5.2.4 Discretized Density Evolution . . . 117
5.3 Density Evolution for the Max-Product Algorithm . . . 122
5.3.1 The Max-Product Algorithm . . . 122
5.3.2 Equivalence of Sum-Product and Max-Product Algorithms . . . 125
5.3.3 Density Evolution for the Max-Product Algorithm . . . 127
5.3.4 The Symmetry Condition . . . 128
5.3.5 Thresholds and Optimization . . . 133
5.4 Summary . . . 135

6 Capacity and Stability **137**

6.1 Channel Examples 137

6.2 Maximum-Stability Functions 140

6.3 Mapping of Thresholds 146

6.4 Summary 155

7 Approximation Methods **157**

7.1 Erasure-Channel Approximation and Optimization 157

7.2 Gaussian Approximation 160

7.2.1 Gaussian Approximation for Regular LDPC Codes 161

7.2.2 Gaussian Approximation for Irregular LDPC Codes 166

7.2.3 Stability 172

7.2.4 Optimization of Degree Distributions using Gaussian Approximation 175

7.2.5 Fixed Points 178

7.3 Gaussian-Capacity Approximation 186

7.4 Reciprocal-Channel Approximation 189

7.4.1 Gaussian and Reciprocal-Channel Approximations 193

7.5 Summary 193

8 LDPC Code Design **197**

8.1 Power-Limited Case 197

8.2 Bandwidth-Limited Case 204

8.3 Summary 207

9 Conclusions **209**

9.1 Thesis Contributions 211

9.2 Future Work 212

A Proofs in Chapter 2 **213**

B Simulation Results for Joint Coding Using Lattice Vector Quantizers **217**

C Simulation Results for Joint Coding Using Approximate Curves **225**

List of Figures

1-1 Separate source and channel coding 25

1-2 Joint source and channel coding 26

2-1 Joint source and channel coding 36

2-2 Linear encoder and decoder 39

3-1 Bandwidth Expansion Using a Two-Dimensional Curve 58

3-2 256-point vector quantizers optimized for the uniform case 63

3-3 SDR vs. SNR plots for the four 256-point vector quantizers in Figure 3-2 for the uniform case. The top solid line is the Shannon limit. 64

3-4 256-point vector quantizers for the Gaussian case. The numerical optimization algorithm (PCCOVQ) of Fuldseth [24] was used. 65

3-5 SDR vs. SNR plots for the four 256-point vector quantizers in Figure 3-4 for the Gaussian case. The top solid line is the Shannon limit. . . . 66

3-6 Generation of the Koch island 68

3-7 Generation of (a) original, (b) switch-back type, and (c) meander type Peano curves 70

3-8 Generation of the (a) 2-dimensional, (b) Moor's version, and (b) 3-dimensional Hilbert curves 71

3-9 Generation of the hexagonal space-filling curve 73

3-10 Encoding for uniform sources. (a) source vector, (b) transmitted offset, (c) received offset. 77

3-11 2:1 bandwidth reduction using the Hilbert curve approximated using a 52-bit number for uniform sources. SDRs for W_1 (—) and W_2 (- -) are plotted as functions of the channel SNR. 81

3-12 Generation of the Hilbert curve mapped on a torus 82

3-13 Two possible patterns for modulo-MMSE decoding for uniform sources on J^2 85

3-14 Stretched curves for Gaussian 88

3-15 2:1 bandwidth reduction using stretched Hilbert curves for Gaussian (number of nodes of each stretched curve shown) 89

3-16 2:1 bandwidth reduction using stretched hexagonal curves for Gaussian (number of nodes of each stretched curve shown) 90

3-17 Joint coding using a uniform spiral curve for two iid unit-variance Gaussians. (a) source vector, (b) transmitted offset, (c) received offset. 92

3-18 2:1 bandwidth reduction using uniform spiral curves for Gaussian . . 94

5-1 A normal graph representation of a low-density parity-check code, where leaf edges are symbols and every edge between an equality constraint on the left hand side and a parity check on the right hand side is a hidden state. 109

5-2 Sum-product decoding for a variable (root node) 112

5-3 Message flow through a variable node 114

5-4 Message flow through a check node 115

5-5 Contour plots of the output message of a degree-3 check node from (a) sum-product and (b) max-product algorithms 124

5-6 The probability of error after one iteration (—) and after two iterations (- -) as functions of the initial probability of error p for a (3,4)-regular LDPC code for the BSC using the max-product algorithm 130

5-7 The probability of error for a (3,4)-regular LDPC code for the BSC using the max-product algorithm 131

5-8 Simulation results for a (3,6) code for the AWGN channel using a block length of 10^6. The sum-product and max-product algorithms are used. 134

6-1 Maximum-stability functions $\xi(r)$ for various channels as functions of the rate r 142

6-2 BSC and AWGN thresholds for various regular ($\circ$) and irregular codes with rates 0.1 to 0.9 designed for AWGN channels with $d_l = 10$ ($\square$) and $d_l = 20$ ($\triangle$). The curves are (from top to bottom) equal-variance, equal-mean, equal-stability, equal-capacity, and equal-SNR. The variances, means, and SNRs are for initial messages. 150

6-3 Laplace and AWGN thresholds for various regular ($\circ$) and irregular codes with rates 0.1 to 0.9 designed for AWGN channels with $d_l = 10$ ($\square$) and $d_l = 20$ ($\triangle$). The curves are (from top to bottom) equal-variance, equal-mean, equal-stability, equal-capacity, and equal-SNR. The variances, means, and SNRs are for initial messages. 151

6-4 BEC and AWGN thresholds for various regular ($\circ$) and irregular codes with rates 0.1 to 0.9 designed for AWGN channels with $d_l = 10$ ($\square$) and $d_l = 20$ ($\triangle$). The curves are (from top to bottom) equal-capacity and equal-stability. ($*$) denotes irregular codes designed for the BEC. ($+$) denotes irregular codes designed for the BEC using the AWGN stability condition. A($\triangle$) is a rate-0.2 irregular code designed for the AWGN channel, B($*$) is a rate-0.2 irregular code designed for the BEC, and C($+$) is a rate-0.2 irregular code designed for the BEC using the AWGN stability condition. All ($*$,$+$) codes have $d_l = 20$. 152

6-5 BEC and BSC thresholds for various regular ($\circ$) and irregular codes with rates 0.1 to 0.9 designed for AWGN channels with $d_l = 10$ ($\square$) and $d_l = 20$ ($\triangle$). The curves are (from top to bottom) equal-capacity and equal-stability. ($*$) denotes an irregular code designed for the BEC. ($+$) denotes an irregular code designed for the BEC using the BSC stability condition. All ($*$,$+$) codes have $d_l = 20$. 153

6-6 BEC and Laplace thresholds for various regular ($\circ$) and irregular codes with rates 0.1 to 0.9 designed for AWGN channels with $d_l = 10$ ($\square$) and $d_l = 20$ ($\triangle$). The curves are (from top to bottom) equal-capacity and equal-stability. 154

7-1 Exact (—), Gaussian approximation (- -), and reciprocal-channel approximation (- ·) thresholds for (j, k)-regular LDPC codes. Thresholds using the reciprocal-channel approximation are not distinguishable from the exact thresholds at this scale. 167

7-2 Approximate (- -) and exact (—) densities at the check node input . 171

7-3 Approximate (- -) and exact (—) densities at the variable node input 172

7-4 $\{h_i(s, r) - r\}$ for $i = 2, \ldots, 20$ (top to bottom) and $\{h(s, r) - r\}$ (- -) 178

7-5 $\{h(s, r) - r\}$ as a function of r magnified 179

7-6 $\{f_i(s, t) - t\}$ for $i = 2, \ldots, 10$ (top to bottom) and $\{f(s, t) - t\}$ (- -) . 180

7-7 $\{f(s, t) - t\}$ as a function of t magnified 181

7-8 Performance of various codes, designed/evaluated using GA/GA, GA/DE, DE/GA, or DE/DE when $d_l = 10$ 182

7-9 Performance of various codes, designed/evaluated using GA/GA, GA/DE, DE/GA, or DE/DE when $d_l = 20$ 183

7-10 The probability of error vs. decrease in the probability of error of the $d_l = 20$ code in Table 8.1 using density evolution (—) and the Gaussian approximation (- -) 184

7-11 The probability of error vs. decrease in the probability of error for the $d_l = 7$ code in (7.18) using density evolution (—) and the Gaussian approximation (- -) 185

7-12 Calculation of the output m_u using reciprocal-channel mappings at a check node 192

7-13 Equivalent calculations under the reciprocal-channel approximation. (a) is the original LDPC code. Parity checks are converted to repetition codes in (b) using reciprocal-channel mappings between the two sides. The dual code of the original LDPC code is used in (c). 192

7-14 Plot of $\alpha = \frac{1}{\psi(x)} \log(1 - \phi(x))$ 194

7-15 $\phi(x)$ (—) vs. $1 - e^{\alpha\psi(x)}$ (- -) using $\alpha = -0.27$. Note that two curves are very close. 195

8-1 Threshold ($\mathrm{SNR}_{\mathrm{norm}}$) of rate-1/2 LDPC codes with maximum variable degree $= 20, \ldots, 8000$ 199

8-2 Simulation results for the $d_l = 100$ and $d_l = 200$ codes of Tables 8.2 and 8.3, using a block length of 10^7. 203

8-3 Multilevel turbo coded modulation 205

8-4 2-level channel example 205

A-1 Test channel. 216

B-1 2:1 bandwidth reduction using the Peano curve for uniform sources. A lattice vector quantizer is used for quantization (number of points of each case shown). SDRs for W_1 (—) and W_2 (- -) are plotted as functions of the channel SNR. 218

B-2 2:1 bandwidth reduction using the Peano curve for uniform sources. A lattice vector quantizer is used for quantization (number of points of each case shown). Average SDRs are plotted as functions of the channel SNR. 219

B-3 2:1 bandwidth reduction using the Hilbert curve for uniform sources. A lattice vector quantizer is used for quantization (number of points of each case shown). SDRs for W_1 (—) and W_2 (- -) are plotted as functions of the channel SNR. 220

B-4 3:1 bandwidth reduction using the three-dimensional Hilbert curve for uniform sources. A lattice vector quantizer is used for quantization (number of points of each case shown). SDRs for W_1 (—), W_2 (- -), and W_3 (- ·) are plotted as functions of the channel SNR. 221

B-5 3:1 bandwidth reduction using the three-dimensional Hilbert curve for uniform sources. A lattice vector quantizer is used for quantization (number of points of each case shown). Average SDRs are plotted as functions of the channel SNR. 222

B-6 2:1 bandwidth reduction using the hexagonal space-filling curve for uniform sources on $\mathcal{R}_V(A_2)$ A lattice vector quantizer is used for quantization (number of points of each case shown). SDRs are plotted as functions of the channel SNR. 223

C-1 2:1 bandwidth reduction using the Peano curve for uniform sources. Approximate curves are used for quantization (number of nodes of each curve shown). SDRs for W_1 (—) and W_2 (- -) are plotted as functions of the channel SNR. 226

C-2 2:1 bandwidth reduction using the Peano curve for uniform sources. Approximate curves are used for quantization (number of nodes of each curve shown). Average SDRs are plotted as functions of the channel SNR. 227

C-3 2:1 bandwidth reduction using a Peano curve of the switch-back type in Figure 3-7 (b) for uniform sources. Approximate curves are used for quantization (number of nodes of each curve shown). SDRs for W_1 (—) and W_2 (- -) are plotted as functions of the channel SNR. 228

C-4 2:1 bandwidth reduction using a Peano curve of the switch-back type in Figure 3-7 (b) for uniform sources. Approximate curves are used for quantization (number of nodes of each curve shown). Average SDRs are plotted as functions of the channel SNR. 229

C-5 2:1 bandwidth reduction using a Peano curve of the meander type in Figure 3-7 (c) for uniform sources. Approximate curves are used for quantization (number of nodes of each curve shown). SDRs for W_1 (—) and W_2 (- -) are plotted as functions of the channel SNR. 230

C-6 2:1 bandwidth reduction using a Peano curve of the meander type in Figure 3-7 (c) for uniform sources. Approximate curves are used for quantization (number of nodes of each curve shown). Average SDRs are plotted as functions of the channel SNR. 231

C-7 2:1 bandwidth reduction using the Hilbert curve for uniform sources. Approximate curves are used for quantization (number of nodes of each curve shown). SDRs for W_1 (—) and W_2 (- -) are plotted as functions of the channel SNR. 232

C-8 2:1 bandwidth reduction using Moor's version of Hilbert's space-filling curve for uniform sources. Approximate curves are used for quantization (number of nodes of each curve shown). SDRs for W_1 (—) and W_2 (- -) are plotted as functions of the channel SNR. 233

C-9 2:1 bandwidth reduction using Moor's version of Hilbert's space-filling curve for uniform sources. Approximate curves are used for quantization (number of nodes of each curve shown). Average SDRs are plotted as functions of the channel SNR. 234

List of Tables

3.1 Peak performance in SDR gap in dB from the Shannon limit for (a) uniform and (b) Gaussian cases, where a 2:1 B/W reduction is achieved using numerically optimized vector quantizers. 62

3.2 The SDR gap in dB from the Shannon limit for uniform sources and 2:1, 3:1 B/W reduction using some space-filling curves. Mid-SNR performance is measured at SNR = 50 dB. Best performance (closest to the Shannon limit) is measured for high SNR > 20 dB. 80

3.3 The SDR gap in dB from the Shannon limit for uniform sources and 2:1 B/W reduction using approximate curves for several space-filling curves. Mid-SNR performance is measured at SNR = 50 dB. Best performance (closest to the Shannon limit) is measured for high SNR > 20 dB. 86

3.4 The minimum SDR gap in dB from the Shannon limit for uniform sources and 2:1 B/W reduction using several approximate curves for the Peano curve. The gap is measured for high SNR > 15 dB. 86

5.1 Quantization effect for a (3,6) code. $[-25, 25]$ and $[-50, 50]$ are used for the ranges of LLR. Number of bits used for quantization versus threshold values in σ and errors in dB relative to the threshold value for 14-bit quantization and for maximum LLR = 25. All thresholds are rounded down. 121

5.2 Quantization effect for the $d_l = 200$ code in Table 8.3. Number of bits used for quantization versus threshold values in σ and errors in dB relative to the threshold value for 14-bit quantization. Maximum LLR = 25 was used. All thresholds are rounded down. 121

5.3 Good rate-1/2 codes with $d_l = 20$ for the sum-product and max-product algorithms for the AWGN channel 134

6.1 Channel capacity at the threshold values of various (j, k)-regular LDPC codes. For each code, threshold values in terms of the channel parameters, ϵ, p, λ, and σ_n, are evaluated using density evolution for the BEC, the BSC, the Laplace and AWGN channels, respectively. These threshold values are used to calculate the channel capacity of the corresponding channels. 147

7.1 Exact and approximate (erasure-channel approximation σ_{ECA}, Gaussian approximation σ_{GA}, and reciprocal-channel approximation σ_{RCA}) threshold values for various (j, k)-regular LDPC codes for the binary-input AWGN channel and sum-product decoding. All threshold values are rounded down. 158

7.2 Exact and approximate (erasure-channel approximation λ_{ECA}, Gaussian approximation λ_{GA}, and reciprocal-channel approximation λ_{RCA}) threshold values for various (j, k)-regular LDPC codes for the binary-input Laplace channel and sum-product decoding. All threshold values are rounded down. 159

7.3 Exact and approximate (erasure-channel approximation p_{ECA}, Gaussian approximation p_{GA}, and reciprocal-channel approximation p_{RCA}) threshold values for various (j, k)-regular LDPC codes for the BSC and sum-product decoding. All threshold values are rounded down. . 159

7.4 Exact and approximate (Gaussian-capacity approximation σ_{GCA}, pairwise Gaussian-capacity approximation σ_{pGCA}, and reciprocal-channel approximation σ_{RCA}) threshold values for various (j,k)-regular LDPC codes for the binary-input AWGN channel and sum-product decoding. All threshold values are rounded down. 188

8.1 Good rate-1/2 codes with $d_l = 20, 30, 40, 50$ 200

8.2 Good rate-1/2 codes with $d_l = 60, 70, 85, 100$ 200

8.3 Good rate-1/2 codes with $d_l = 200, 500, 1000$ 201

8.4 Good rate-1/2 codes with $d_l = 2000, 4000, 8000$ 202

8.5 Good rate-1/2 codes with $d_l = 20$ designed for AWGN, mod-AWGN, the highest level of 4-PAM, and AWGN ($d_l = 4000$), from the second to the fifth columns, respectively. Threshold values for AWGN, mod-AWGN, and PAM channels are given in terms of $\mathrm{SNR}_{\mathrm{norm}}$ in dB. All SNR values are rounded up. 207

Chapter 1

Introduction

1.1 Motivation

Suppose we have a noisy communication channel through which we want to send information reliably. Shannon [59] showed that arbitrarily reliable transmission is possible through this channel if the information rate in bits per channel use is less than the *channel capacity* of the channel.

Shannon also showed that there exists a minimum number of bits needed to represent an independent and identically distributed (iid) source within a given fidelity criterion. Figure 1-1 shows this two-stage processing of information, called source and channel coding. Since bits are a universal form of information that do not depend on source or channel characteristics, this separate source-channel coding ensures that we can design two codes separately, i.e., the design of source code does not depend on the channel and vice versa. Furthermore, Shannon proved that this separation is without loss of optimality.

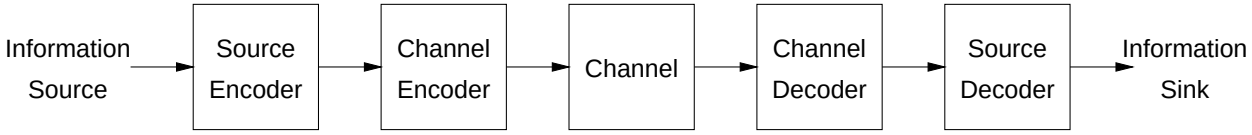


Figure 1-1: Separate source and channel coding

In practice, however, mainly due to channel impairment, we sometimes need to

adjust the amount of information flow from the source encoder to the channel encoder. This can be done by designing variable-rate source and channel codes and adjusting their rates as the channel changes. If the channel changes rapidly, however, this is not a good idea since separate source and channel codes usually require a large block length for good performance, and we may not be able to adjust the source and channel codes quickly enough.

This thesis proposes various novel joint source and channel coding schemes that use the channel only once. Since there is no coding delay, these schemes are useful when the channel statistics change extremely rapidly. By employing several memoryless analog coding schemes, this thesis shows the Shannon limit can be approached very closely for many interesting sources and channels. In some cases, when the source bandwidth is equal to the channel bandwidth, these schemes actually achieve the Shannon limit for all values of SNR. This thesis proposes using space-filling curves and other families of curves when two bandwidths are different. We show that, in many cases, the Shannon limit can be approached very closely. Our bandwidth-reducing schemes are about 1 dB from the rate-distortion bound, which is quite good considering that the channel is used only once.



Figure 1-2: Joint source and channel coding

To approach the Shannon limit further, however, we need to use coding. Recently rediscovered low-density parity check (LDPC) codes are currently the best-known practically implementable codes for approaching the channel capacity of various channels very closely. This thesis proposes very fast and accurate algorithms to optimize LDPC codes, and also proposes several tools to analyze the sum-product algorithm for LDPC codes. We have designed LDPC codes for AWGN channels that approach the Shannon limit very closely. For binary-input AWGN channels, our best code is within 0.0045 dB of the Shannon limit asymptotically, and is within 0.04 dB of the Shannon limit at a bit error rate of 10^{-6} when a block length of 10^7 is used. Our

asymptotic result is more than 10 times closer to the Shannon limit than the previous record by Richardson *et al.* [53]. Our simulation result is more than 3 times closer to the Shannon limit than the previous record in [53].

Interestingly, these two methods operate in opposite extreme regions. Our joint source and channel coding schemes have a block length of one and the LDPC codes have a block length approaching infinity. By concentrating on these two extreme regions, which makes design and analysis easier, we have been able to push the performance of each coding scheme extremely close to the Shannon limit.

1.2 Joint Source and Channel Coding

Modern digital transmission of analog signals such as audio or video through analog channels is rendering analog modulation schemes obsolete. For example, it is now cheaper and more bandwidth efficient to compress a voice signal and send it through a telephone line using only a fraction of the bandwidth required to send the analog voice signal directly through the phone line. The same is true for the digital transmission of multimedia signals such as HDTV broadcasting, digital audio broadcasting, and digital image recording on magnetic media. However, this benefit comes from the fact that such analog sources are highly compressible, not necessarily because digital transmission is always the most efficient. To demonstrate this, we will show that there are many practical examples of sources and channels where simple analog processing can achieve Shannon's rate-distortion limit without any coding.

Another shortcoming of pure digital transmission of analog signals through an analog channel is that it exhibits on-off behavior; i.e., the distortion becomes very large if the channel noise becomes larger than nominal, while there is no improvement in the quality if the channel noise is smaller than nominal. This is why some form of joint source-channel coding, where source and channel codes are changed together according to the channel noise, is used in many digital transmission schemes. We show that in many useful cases it is possible to construct an analog transmission scheme that is optimal, or near-optimal, for a wide range of channel SNR without any adjustments

when the channel parameters change. Such schemes would be particularly useful when the channel is highly non-stationary and the SNR changes rapidly.

When the channel is stationary, Shannon's separability theorem [59] guarantees no loss of optimality when we design source and channel codes separately. However, with such separation, infinite delay and infinite complexity are required to achieve the Shannon limit. Moreover, the separation theorem breaks down in some cases such as multiuser channels [13]. Finally, there are situations in which combined source-channel schemes are much simpler, such as the following well-known example: if an iid Gaussian source and an AWGN channel have the same bandwidth, then a simple uncoded system combined with a linear least-square error (LLSE) receiver can achieve Shannon's rate-distortion bound for any given value of SNR [30]. In this thesis we will construct another interesting example of source and channel in which an uncoded scheme achieves optimality for all values of SNR.

It is generally impossible to approach optimality arbitrarily closely for two different values of SNR with a single separate coding scheme. For example, it has been shown that no single modulation system that maps an analog source vector to a continuous AWGN channel input can achieve the rate-distortion bound on the mean-square error for all values of SNR if the channel bandwidth is greater than the source bandwidth [76].

To approach the optimality for a range of SNR via separate source-channel coding, we need to construct a parametrized pair of source and channel codes and adjust the codes adaptively as the channel SNR varies. Many practical schemes, too numerous to mention, fit into this category. Without this adaptation, a separate source-channel coding scheme generally exhibits a flat performance up to a certain SNR and breaks down for lower values of SNR. This occurs because the source coder does not have any information about the channel parameters. For this reason, a joint source-channel coding scheme that shows graceful degradation is sometimes desirable for non-stationary channels.

One approach to this problem is to find an optimal mapping between source and channel codebooks, where two codebooks are either jointly or separately optimized.

Several good algorithms exist to numerically design a vector quantizer in the presence of channel noise [18, 65]. This was generalized to the case when the channel input is power constrained in [25, 24]. These algorithms can also be used when the source and channel bandwidths are different. However, due to the complexity and the stability of the algorithm, the codebook size is limited. We provide analysis of some of the results in [24] for Gaussian channels and show it is possible to find optimal coding schemes analytically when the coding scheme is restricted to spiral or spiral-like curves.

Shannon noted some examples of joint source-channel coding schemes in one of his original papers [60]. He also described a threshold effect in bandwidth-expanded transmissions where a certain amount of noise does not affect the system performance much until the noise reaches a critical level. A more detailed analysis of bandwidth expansion via n-dimensional curves was done by Wozencraft and Jacobs [74].

In modern digital transmission systems, however, the greater concern is the efficient use of bandwidth. In this thesis, we use space-filling curves for bandwidth reduction for several types of continuous sources and continuous noisy channels, and show how to construct robust joint source-channel coding schemes in the presence of channel impairments.

1.3 Low-Density Parity-Check Codes

If we want to approach the Shannon limit very closely, we can use separate source-channel coding. Although it is still difficult to approach the rate-distortion bound for an arbitrary source, because it is usually hard to characterize such a source, it is relatively easier to approach the channel capacity of many useful channels. Therefore, in this thesis, we concentrate only on the channel coding and show how to approach the Shannon limit of the additive white Gaussian noise (AWGN) channel using low-density parity-check (LDPC) codes.

LDPC codes were invented by Gallager in his thesis [26, 27]. The term low-density means that the number of 1's in each row and column of the parity check matrix is small compared to the block length. LDPC codes did not get much attention for

many decades until recently when highly successful turbo codes were discovered [4]. LDPC codes were then rediscovered by Spielman *et al.* [62] and MacKay *et al.* [46].

For many channels and iterative decoders of interest, low-density parity-check (LDPC) codes exhibit a threshold phenomenon [54]: as the block length tends to infinity, an arbitrarily small bit error probability can be achieved if the noise level is smaller than a certain threshold. For a noise level above this threshold, on the other hand, the probability of bit error is larger than a positive constant. Gallager first observed this phenomenon for the binary symmetric channel (BSC) when he introduced regular LDPC codes [26, 27] using an explicit construction of regular graphs. Luby *et al.* generalized this idea to randomly constructed irregular LDPC codes, showed that irregular codes perform better than regular ones, and also showed that the threshold phenomenon occurs for these codes [44].

In [54], this observation was further generalized by Richardson and Urbanke to a large range of binary-input channels, including binary erasure, binary symmetric, Laplace, and AWGN channels, and to various decoding algorithms including belief propagation (sum-product algorithm), which are collectively called *message-passing* algorithms. Richardson *et al.* proved a general concentration theorem showing that the decoder performance on random graphs converges to its expected value as the length of the code increases, generalizing the result of Luby *et al.* [44]. Since it is difficult to determine the expected performance for an ensemble of finite size, they used the expected behavior in the limit of infinitely long codes, which can be determined from the corresponding cycle-free graph. They defined the threshold as indicated above for a random ensemble of irregular codes specified by degree distributions, and developed an algorithm called *density evolution* for iteratively calculating message densities, enabling the determination of thresholds.

Using this result, they constructed LDPC codes that clearly beat the powerful turbo codes [4] on AWGN channels. Recently, this was improved in [8], suggesting that LDPC codes might approach the channel capacity of the AWGN channel asymptotically.

Calculating thresholds and optimizing degree distributions using density evolution

is a computationally intensive task for most channels other than BECs. In BECs, density evolution becomes one-dimensional, and it is possible to do more analysis and even to construct capacity-achieving codes [45]. For more interesting channels, including AWGN channels, however, density evolution is too complicated to be analyzed.

In this thesis, we propose four novel approximation methods to estimate the threshold of LDPC codes on various binary-input symmetric-output memoryless channels with sum-product decoding. These approximation methods are based on tracking the behavior of density evolution reliably using a one-dimensional representative of a message density.

The first method, erasure-channel approximation, is based on the observation that threshold values for one channel can be mapped to threshold values for other channels with good accuracy using a certain mapping. The second method, Gaussian approximation, is based on approximating message densities as Gaussians (for regular LDPC codes) or Gaussian mixtures (for irregular LDPC codes) [12]. The third method, Gaussian-capacity approximation, is based on the interpretation that density evolution is equivalent to iterative channel normalization. The forth method, reciprocal-channel approximation, which is based on dualizing LDPC codes, provides a very accurate model of density evolution for AWGN channels.

Since these methods are easier to analyze and are computationally faster than density evolution, they can be useful tools for understanding the behavior of the decoder and for optimizing irregular codes. For example, we show how easily optimization can be done using erasure-channel and Gaussian approximations. We also show how to determine the rate of convergence of the error probability, and also show why there is an alternation between fast and slow decoding stages, as noted in [53]. We also use these methods to design good irregular codes using linear programming. These algorithms optimize degree distributions several orders of magnitude faster and they are often as good as the optimization methods based on the full density evolution.[1]

We also present some tools to analyze density evolution for the max-product

[1]To design codes extremely close to the Shannon limit, we need to use density evolution.

algorithm, which can be viewed as a simplified version of the sum-product algorithm because it is simpler to decode and it does not require the estimation of the channel noise for the AWGN channel. We show several examples where we lose about 0.6 dB for the AWGN channel when the max-product algorithm is used instead of the sum-product algorithm. We also show that codes need to be optimized for the max-product algorithm if we want to approach the max-product capacity.

We find a rough measure of how easy or difficult it is to design LDPC codes for binary-input symmetric-output channels. We prove that the BSC is the most difficult and the BEC is the most easiest to design LDPC codes for in this sense.

This thesis also proposes a very fast and accurate algorithm to compute thresholds for various channels. In this thesis, an optimization algorithm for degree distributions is proposed that can push the performance of LDPC codes to within 0.0045 dB of the Shannon limit of binary-input AWGN channels, which is more than 10 times closer to the Shannon limit than the previous record by Richardson *et al.* [53]. Our simulation results show that we can actually operate within 0.04 dB of the Shannon limit at a bit error rate of 10^{-6} using a block length of 10^7. This is more than 3 times closer to the Shannon limit than the previous record in [53].

In case of bandwidth-limited AWGN channels, we show how to design LDPC codes for some multilevel coding schemes. By using the proposed optimization algorithm for each level, we construct LDPC codes that are within 0.0063 dB of the noisiest level of some pulse amplitude modulation (PAM) channels.

1.4 Thesis Organization

In Chapter 2, we demonstrate that Shannon's rate-distortion limit can be achieved using some uncoded systems. In Section 2.1, we give the source and channel model for our joint source and channel coding schemes. In Section 2.2, we define the channel capacity and the rate-distortion function for our source and channel model. In Section 2.3, we define several types of matched sources and channels and give some examples. In Section 2.4, we define a uniform source with modulo distortion and find

its rate-distortion function. In Section 2.5, we define a modulo channel and find its channel capacity. In Section 2.6, we show the optimality of an uncoded system for this uniform source and the modulo channel when the source and the channel have the same bandwidth.

In Chapter 3, we use space-filling curves and other families of curves to construct efficient joint source-channel coding schemes when the source bandwidth is greater than the channel bandwidth. In Section 3.1, we show how to use curves for bandwidth reduction in general. We explain space-filling curves in Section 3.2. We use space-filling curves to construct near-optimal coding schemes for the uniform source and the modulo channel in Section 3.3. We generalize this idea to a Gaussian source and the AWGN channel in Section 3.4, and construct some joint coding schemes to approach the Shannon limit very closely.

From Chapters 4 to 8, we consider how to analyze density evolution and how to design good LDPC codes.

In Chapter 4, we introduce binary-input symmetric-output memoryless channels. In Section 4.2, we show how to normalize these channels in a unified approach.

In Chapter 5, we introduce LDPC codes and two decoding algorithms. We introduce codes on graphs and normal realizations of LDPC codes in Section 5.1. In Section 5.2, we introduce the sum-product algorithm and its density evolution. We show the relationship between density evolution and iterative normalization. We also show how to implement a very accurate density evolution algorithm. In Section 5.3, we show some interesting aspects of density evolution for the max-product algorithm.

In Chapter 6, we develop a tool to compare various symmetric channels. In Section 6.1, we give several examples of symmetric channels. In Section 6.2, we define maximum-stability functions for symmetric channels and show their properties. In Section 6.3, we show some empirical observations on threshold values for different channels and discuss the implications, which will be later used as motivations for developing some of approximation methods in Chapter 7.

In Chapter 7, we develop four approximation methods for analyzing density evolution for the sum-product algorithm. In Section 7.1, we present the erasure-channel

approximation and optimization methods based on the approximation. In Section 7.2, we construct a Gaussian approximation method and present some analysis of density evolution. In Section 7.3, we construct a Gaussian-capacity approximation. In Section 7.4, we show how to develop a very accurate model of the density evolution for AWGN channels, which is called the reciprocal-channel approximation.

In Chapter 8, we design capacity-approaching LDPC codes. In Section 8.1, we design LDPC codes for power-limited AWGN channels. In Section 8.2, we design LDPC codes for bandwidth-limited AWGN channels.

In Chapter 9, we summarize the contributions of this thesis and propose future research.

The material in Chapter 3 was presented in part at [11]. The material in Chapters 2 and 3 is in preparation for publication [10].

The material in Section 7.2 was presented in part at [12], and was submitted to [9]. The material in Sections 5.2.4 and 8.1 was submitted in part to [8]. The material in Chapters 4 and 6 and Sections 5.2.3, 7.1, 7.3, and 7.4 is in preparation for publication [7].

Chapter 2

Matched Sources and Channels

Pilc [52] and Jelinek [38] defined that a source and channel pair is matched if the optimal distortion is achieved independent of the block length or, in other words, without any coding.

In this chapter, we define three types of matchedness to further categorize matched sources and channels. We also define a uniform source on a fundamental region of a lattice with a modulo-difference-distortion measure. We show that this source and a modulo additive white noise channel with a certain corresponding noise density are matched.

We first define our source and channel model to be used in Chapters 2 and 3.

2.1 Source and Channel Model

Figure 2-1 shows a block diagram of a joint source-channel coding system to be used in Chapters 2 and 3. The source $\mathbf{W}$ with probability density function (pdf) $p_{\mathbf{W}}(\mathbf{w})$ and the reproduction alphabet $\hat{\mathbf{W}}$ are defined on $\mathcal{W}$ and $\hat{\mathcal{W}}$, respectively. The channel input $\mathbf{X}$ and the output $\mathbf{Y}$ are defined on $\mathcal{X}$ and $\mathcal{Y}$, respectively. The channel output $\mathbf{Y}$ is related to the channel input $\mathbf{X}$ by a transition probability $p_N(\mathbf{y}|\mathbf{x})$, where $N \in \mathcal{N}$ is the noise parameter of the channel. The encoder is a mapping $f(\cdot)$ from $\mathcal{W}$ to $\mathcal{X}$, and the decoder is a mapping $g(\cdot)$ from $\mathcal{Y}$ to $\hat{\mathcal{W}}$.

We call n the dimension of a set $\mathcal{A}$ if $\mathcal{A} \subset \mathcal{U}^n$ and if no $n' < n$ satisfies $\mathcal{A} \subset \mathcal{U}^{n'}$,



Figure 2-1: Joint source and channel coding

where $\mathcal{U}$ is either $\mathbb{R}$, $\mathbb{C}$, or a finite set. We further assume that the dimensions of $\mathcal{W}$ and $\hat{\mathcal{W}}$ are n and the dimensions of $\mathcal{X}$ and $\mathcal{Y}$ are m.[1]

Let $d(\cdot,\cdot)$ be a distortion measure that maps $\mathcal{W} \times \hat{\mathcal{W}}$ to $\mathbb{R}^+$, which we assume bounded, i.e.,

$$\sup_{\mathbf{w}\in\mathcal{W},\, \hat{\mathbf{w}}\in\hat{\mathcal{W}}} d(\mathbf{w}, \hat{\mathbf{w}}) < \infty.$$

Let $s(\mathbf{x})$ denote a function that acts as a constraint on the input distribution $p(\mathbf{x})$ of $\mathbf{X}$, i.e., $E[s(\mathbf{X})] \leq P$, where P is a given parameter (power). We give some examples in the following.

Example 2.1 (A zero-mean Gaussian source with a squared distortion measure) *The source and reproduction spaces are $\mathbb{R}$, and a squared distortion measure is defined as $d(w,\hat{w}) = ||w-\hat{w}||^2$. The source has a zero-mean Gaussian density with variance S.*

Example 2.2 (A uniform source with a Hamming distortion) *The source and reproduction spaces are $\mathcal{W} = \hat{\mathcal{W}} = \{0, 1, \ldots, k-1\}$, and a Hamming distortion is defined as*

$$d(w,\hat{w}) = \begin{cases} 0, & \text{if } w = \hat{w}; \\ 1, & \text{otherwise.} \end{cases}$$

The source is uniformly distributed on $\mathcal{W}$.

Example 2.3 (An AWGN channel with power constraint) *The channel input and output spaces are $\mathbb{R}$. The constraint function is defined as $s(x) = ||x||^2$. The*

[1]For example, the input and output dimensions of the binary-input AWGN channel are one.

channel output Y is given by

$$Y = X + Z,$$

where Z is a zero-mean Gaussian with variance N that is independent of the input.

Example 2.4 (BSC) *The channel input and output spaces are $\{0, 1\}$. There is no power constraint for this channel. The input is complemented with probability $p \in [0, \frac{1}{2}]$.*

Our problem is to find the optimal encoder and decoder pair, i.e., one that minimizes the expected distortion $E[d(\mathbf{W}, \hat{\mathbf{W}})]$ subject to the power constraint $E[s(\mathbf{X})] \leq P$. To measure the performance of our coding schemes, we will use the Shannon limit as a reference, which is given in the following section.

2.2 Shannon Limit

If the encoder and decoder are allowed to process a block of inputs rather than a single input at a time, then we can use coding. In this case, we assume the source sequence $\{\mathbf{W}_i | 1 \leq i \leq k\}$ is independent and identically distributed (iid) and the memoryless channel is used a certain number of times, where k is the block length.

In this case, a lower bound on the distortion is given by Shannon's source-channel separability theorem.

The rate-distortion function $R(D)$ of the source $\mathbf{W}$ with a distortion measure $d(\mathbf{w}, \hat{\mathbf{w}})$ is given by

$$R(D) = \min_{p(\hat{\mathbf{w}}|\mathbf{w}):\, \int d(\mathbf{w},\hat{\mathbf{w}}) p(\hat{\mathbf{w}}|\mathbf{w}) p_{\mathbf{W}}(\mathbf{w})\, d\mathbf{w}\, d\hat{\mathbf{w}} \leq D} I(\mathbf{W}; \hat{\mathbf{W}}).$$

where $R(D)$ is the minimum number of bits per symbol required to achieve the expected distortion to be less than or equal to D. Note that $R(D)$ is a monotonically nonincreasing function of D.

The channel capacity of the channel $p_N(\mathbf{y}|\mathbf{x})$ is given by

$$C(N) = \max_{p(\mathbf{x}):\, \int s(\mathbf{x})p(\mathbf{x})\, d\mathbf{x} \le P} I(\mathbf{X};\mathbf{Y}),$$

where $C(N)$ is the maximum number of bits per symbol we can transmit reliably through the channel.

Shannon's source-channel separability theorem states that a two-stage system with separate source and channel coding is without loss of optimality. Therefore, the expected distortion achievable by any coding scheme is lowerbounded by the following:

$$D_{\text{opt}} = \inf\{D : R(D) < C(N)\}.$$

2.2.1 Optimal Decoder

Let us assume that an encoder $f(\cdot)$ is given. The optimal decoder $\hat{g}(\cdot)$ is a decoder that minimizes the expected distortion $E[d(\mathbf{W}, g(\mathbf{Y}))]$, i.e.,

$$\hat{g}(\cdot) = \operatorname*{argmin}_{g(\cdot)} E[d(\mathbf{W}, g(\mathbf{Y}))]$$

Since $E[d(\mathbf{W}, g(\mathbf{Y}))]$ can be rewritten as $E_{\mathbf{y}}[E_{\mathbf{w}|\mathbf{y}}[d(\mathbf{W}, g(\mathbf{Y}))|\mathbf{Y} = \mathbf{y}]]$, it is enough to find the $g(\mathbf{y})$ that minimizes $E_{\mathbf{w}|\mathbf{y}}[d(\mathbf{W}, g(\mathbf{Y}))|\mathbf{Y} = \mathbf{y}]$ for every $\mathbf{y} \in \mathcal{Y}$.

When a squared distortion measure is used, the problem reduces to a minimum mean-square error (MMSE) estimation.

2.2.2 Optimal Encoder

Finding the optimal encoder is usually more difficult than finding the optimal decoder.

However, if we impose certain constraints in the encoder, then it sometimes becomes easier to find optimal encoders. For example, when the source and the channel are analog and both encoder and decoder are linear, it is possible to find the optimal encoder and decoder pair.

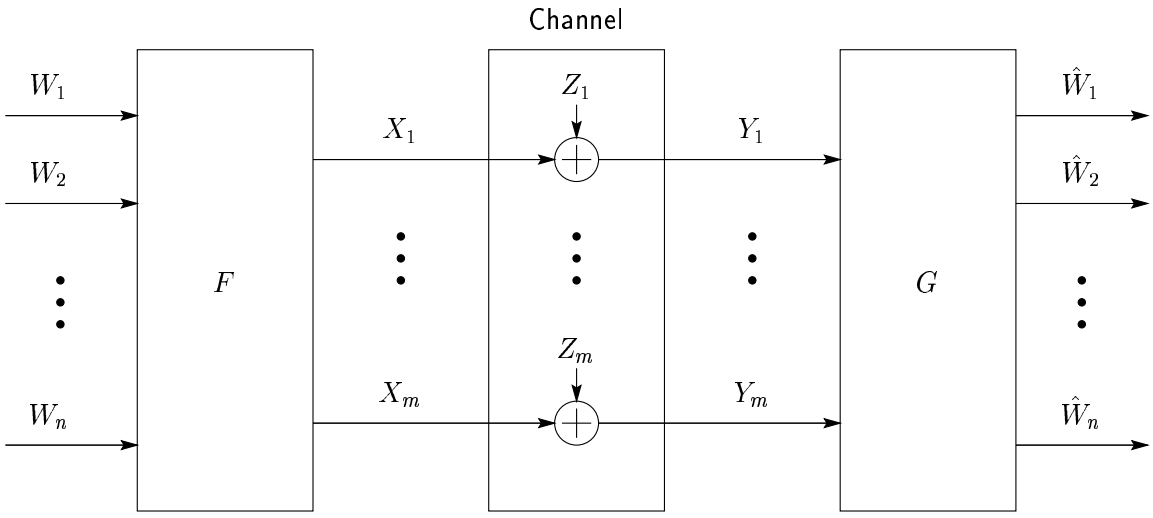


Figure 2-2: Linear encoder and decoder

Linear Encoder and Decoder

We assume the source $\mathbf{W}$ and the reproduction $\hat{\mathbf{W}}$ are n-dimensional real vectors. We assume an additive-noise channel, where the channel input $\mathbf{X}$, the output $\mathbf{Y}$, and the additive noise $\mathbf{Z}$ are m-dimensional real vectors and $\mathbf{Z}$ is independent of $\mathbf{W}$. We assume a power constraint on the channel input, i.e.,

$$E[||\mathbf{X}||^2] \leq P.$$

We also assume that the encoder and the decoder are linear, i.e., the encoder becomes a matrix F and the decoder becomes a matrix G as in Figure 2-2. The decoder output $\hat{\mathbf{w}}$ can be written as

$$\hat{\mathbf{w}} = G(F\mathbf{w} + \mathbf{z}).$$

Now, the problem of finding the optimal F and G that minimizes the expected squared distortion is given by the following:

$$\begin{aligned} \min_{F,G} \quad & \left\{ E[||\hat{\mathbf{W}} - \mathbf{W}||^2] = \operatorname{tr}(GF - I)\Lambda_w(F'G' - I) + \operatorname{tr} G\Lambda_z G' \right\} \\ \text{subject to} \quad & E[||F\mathbf{W}||^2] = \operatorname{tr} F\Lambda_w F' \leq P. \end{aligned} \tag{2.1}$$

The matrix equation (2.1) was solved analytically in [41, 42].[2] The optimal encoder is composed of three stages. It first uncorrelates the source into n independent components by diagonalizing Λ_w. It also diagonalizes the noise covariance matrix Λ_z at the last stage, so that the second stage of the encoder sees independent sources and independent channels. The optimal strategy for the second stage is to allocate the the strongest source component to the least noisy channel component, the second strongest source to the next least noisy channel, and so on. Gains need to be optimized for each mapping from the source to the channel components.

As a simple example, when the channel dimension m is one, then the optimal encoder is the eigenvector corresponding to the largest eigenvalue of Λ_w times a constant, and the optimal decoder is the same vector, transposed and scaled.

When the source dimension n is one, then the optimal encoder is the eigenvector corresponding to the smallest eigenvalue of Λ_z times a constant, and the optimal decoder is the same vector, transposed and scaled.

Permutation

As a special case of the linear encoder and decoder, we consider permutation matrices for the encoder and decoder. We assume $n = m$. In this case, the power constraint is omitted since we are not allowed to control the transmitted power. We also assume that the source and channel are ordered, i.e., $\sigma_{w_i}^2 > \sigma_{w_j}^2$ for $1 \leq i < j \leq n$ and $\sigma_{z_i}^2 < \sigma_{z_j}^2$ for $1 \leq i < j \leq n$. Let $\{j_i |\ 1 \leq i \leq n\}$ be a set of indices, where the i-th source is mapped to the j_i-th channel.

The optimal decoder is the inverse permutation followed by a set of n linear least-square estimators. In this case, the expected distortion D becomes

$$D = \sum_{i=1}^{n} \frac{\sigma_{w_i}^2 \sigma_{z_{j_i}}^2}{\sigma_{w_i}^2 + \sigma_{z_{j_i}}^2}.$$

We prove that the optimal permutation encoder is an identity mapping, i.e., more powerful signals are assigned to less noisy channels. Note that this is similar to

[2] They also considered individual power constraints.

combining water-pouring and reverse water-filling arguments in parallel Gaussian channels and independent Gaussian sources.

Lemma 2.1 *The optimal permutation encoder is an identity mapping.*

Proof: Let us assume that $j_i > j_k$ for some $i < k$. Then,

$$\begin{aligned}
&\left(\frac{\sigma_{w_i}^2\sigma_{z_{j_i}}^2}{\sigma_{w_i}^2+\sigma_{z_{j_i}}^2}+\frac{\sigma_{w_k}^2\sigma_{z_{j_k}}^2}{\sigma_{w_k}^2+\sigma_{z_{j_k}}^2}\right)-\left(\frac{\sigma_{w_i}^2\sigma_{z_{j_k}}^2}{\sigma_{w_i}^2+\sigma_{z_{j_k}}^2}+\frac{\sigma_{w_k}^2\sigma_{z_{j_i}}^2}{\sigma_{w_k}^2+\sigma_{z_{j_i}}^2}\right)\\
&\quad=\frac{(\sigma_{z_{j_k}}^2-\sigma_{z_{j_i}}^2)(\sigma_{w_k}^2-\sigma_{w_i}^2)\{\sigma_{z_{j_i}}^2\sigma_{z_{j_k}}^2(\sigma_{w_i}^2+\sigma_{w_k}^2)+\sigma_{w_i}^2\sigma_{w_k}^2(\sigma_{z_{j_i}}^2+\sigma_{z_{j_k}}^2)\}}{(\sigma_{w_i}^2+\sigma_{z_{j_i}}^2)(\sigma_{w_k}^2+\sigma_{z_{j_k}}^2)(\sigma_{w_i}^2+\sigma_{z_{j_k}}^2)(\sigma_{w_k}^2+\sigma_{z_{j_i}}^2)}\\
&\quad>0.
\end{aligned}$$

□

2.3 Matched Sources and Channels

In this section, we find conditions for optimal joint source-channel coding schemes when the source and channel bandwidths are equal.

Shannon's source-channel separability theorem says that a two-stage source-channel coding system is as good as any one-stage system in which the encoder maps the source vector to the channel input directly. But, is there any one-stage coding scheme that is optimal for all $N \in \mathcal{N}$ without using a large block length? The answer is yes. We will characterize some optimal coding schemes as follows.

We first define matched sources and channels in which case optimality is achieved by sending the source through the channel using an identity mapping and by a decoder that is also an identity mapping.

From now on, we assume that the source alphabet $\mathcal{W}$ is the same as the channel input alphabet $\mathcal{X}$ and that the channel output alphabet $\mathcal{Y}$ is the same as $\hat{\mathcal{W}}$.

Definition 2.1 *A channel $p_N(\mathbf{y}|\mathbf{x})$ with a constraint function $s(\mathbf{x})$ and a source $\mathbf{W}$ with pdf $p_{\mathbf{W}}(\mathbf{w})$ and a distortion measure $d(\cdot,\cdot)$ are* matched *if*

1. $\int p_{\mathbf{W}}(\mathbf{w}) s(\mathbf{w})\, d\mathbf{w} \leq P$,

2. $C(N) = R(D(N))$ *for all* $N \in \mathcal{N}$,

where $C(N)$ *is the channel capacity of the channel,* $R(D)$ *is the rate-distortion function of the source, and*

$$D(N) = \int p_N(\hat{\mathbf{w}}|\mathbf{w}) p_{\mathbf{W}}(\mathbf{w}) d(\mathbf{w}, \hat{\mathbf{w}})\, d\mathbf{w}\, d\hat{\mathbf{w}}$$

is the expected distortion when the channel input $\mathbf{X}$ *is* $\mathbf{W}$ *and the channel output* $\mathbf{Y}$ *is* $\hat{\mathbf{W}}$.

When a source and a channel are matched, an uncoded scheme yields the expected distortion $D(N)$, which is optimal for all $N \in \mathcal{N}$. Unfortunately, it is not clear how to find such matched sources and channels.

In the following, we will define other types of matched sources and channels, which will be more useful for finding matched sources and channels. We use the same notation $D(N)$ as in the previous definition to denote the expected distortion when an uncoded scheme is used.

Definition 2.2 *A channel* $p_N(\mathbf{y}|\mathbf{x})$ *with a constraint function* $s(\mathbf{x})$ *and a source* $\mathbf{W}$ *with pdf* $p_{\mathbf{W}}(\mathbf{w})$ *and a distortion measure* $d(\cdot, \cdot)$ *are* entropy-matched *if*

1. $\int p_{\mathbf{W}}(\mathbf{w}) s(\mathbf{w})\, d\mathbf{w} \;\leq\; P$,

2. $h(\mathbf{Y}|\mathbf{X} = \mathbf{x})$ *does not depend on* $\mathbf{x}$,

3. $$\max_{p(\mathbf{x}):\, \int s(\mathbf{x}) p(\mathbf{x})\, d\mathbf{x} \leq P} h(\mathbf{Y}) = h(\mathbf{W}),\ \forall N \in \mathcal{N},$$

4. $$\max_{p(\hat{\mathbf{w}}|\mathbf{w}):\, \int d(\mathbf{w},\hat{\mathbf{w}}) p(\hat{\mathbf{w}}|\mathbf{w}) p_{\mathbf{W}}(\mathbf{w})\, d\mathbf{w}\, d\hat{\mathbf{w}} \leq D(N)} h(\mathbf{W}|\hat{\mathbf{W}}) = h(\mathbf{Y}|\mathbf{X}),\ \forall N \in \mathcal{N},$$

where $h(\cdot)$ is the entropy or the differential entropy function, depending on the type of the argument.

It is easy to check that if a matched source and channel pair satisfies conditions 2 and 3 above, then it is entropy matched.

For the following definition, we assume that not only $\mathcal{W} = \mathcal{X}$ and $\hat{\mathcal{W}} = \mathcal{Y}$, but also that all four spaces are the same; i.e., $\mathbf{W}, \hat{\mathbf{W}}, \mathbf{X}, \mathbf{Y}$ are all defined on the same space $\mathcal{W}$.

Definition 2.3 *A channel $p_N(\mathbf{y}|\mathbf{x})$ with a constraint function $s(\mathbf{x})$ and a source $\mathbf{W}$ with pdf $p_{\mathbf{W}}(\mathbf{w})$ and a distortion measure $d(\cdot,\cdot)$ are* density-matched *if*

1. $\int p_{\mathbf{W}}(\mathbf{w}) s(\mathbf{w})\, d\mathbf{w} \;\leq\; P$,
2. $h(\mathbf{Y}|\mathbf{X} = \mathbf{x})$ *does not depend on* $\mathbf{x}$,
3. $$\operatorname*{argmax}_{p(\mathbf{x}):\, \int s(\mathbf{x}) p(\mathbf{x})\, d\mathbf{x} \leq P} h(\mathbf{Y}) = p_{\mathbf{W}}(\mathbf{x}),\ \forall N \in \mathcal{N},\ \textit{almost everywhere}$$
4. $$p_{\mathbf{W}}(\mathbf{w}) \left\{ \operatorname*{argmax}_{p(\hat{\mathbf{w}}|\mathbf{w}):\, \int d(\mathbf{w},\hat{\mathbf{w}}) p(\hat{\mathbf{w}}|\mathbf{w}) p_{\mathbf{W}}(\mathbf{w})\, d\mathbf{w}\, d\hat{\mathbf{w}} \leq D(N)} h(\mathbf{W}|\hat{\mathbf{W}}) \right\} = p_N(\mathbf{w}|\hat{\mathbf{w}}) p_{\mathbf{W}}(\hat{\mathbf{w}}),\ \forall N \in \mathcal{N},\ \textit{almost everywhere},$$

where $p_{\mathbf{W}}(\cdot)$ is the pdf of $\mathbf{W}$.

The following lemma shows an immediate application of the above definition, which is merely an interpretation of condition 4 of the above definition.

Lemma 2.2 *If a channel $p_N(\mathbf{y}|\mathbf{x})$ with a constraint function $s(\mathbf{x})$ and a source $\mathbf{W}$ with a distortion measure $d(\cdot,\cdot)$ are density matched, then the channel achieves the minimum mutual information between $\mathbf{W}$ and $\hat{\mathbf{W}}$ for a given distortion $D(N)$, where $\hat{\mathbf{W}} \sim p_{\mathbf{W}}(\hat{\mathbf{w}})$ is the input and $\mathbf{W}$ is the output of the channel.*

Therefore, for a density-matched source and channel, the test channel that achieves the minimum mutual information between the source and the reproduction can be constructed using the original channel with the input distribution $p_{\mathbf{W}}$.

The following lemma shows how the above three matchedness are related.

Lemma 2.3 *If a channel $p_N(\mathbf{y}|\mathbf{x})$ with a constraint function $s(\mathbf{x})$ and a source $\mathbf{W}$ with a distortion measure $d(\cdot,\cdot)$ are density matched, then they are entropy matched, but the converse is not true. If they are entropy matched, then they are matched, but the converse is not true.*

Proof: If a source and a channel are entropy matched, then it is easy to verify that they are matched. The converse is not true as shown in Example 2.7 in the following section.

If a source and channel pair is density matched, then the joint distribution $p(\mathbf{w}, \hat{\mathbf{w}})$ of $\mathbf{W}$ and $\hat{\mathbf{W}}$ that maximizes $h(\mathbf{W}|\hat{\mathbf{W}})$ in Definition 2.3 is equal to $p_N(\mathbf{w}|\hat{\mathbf{w}})p_{\mathbf{W}}(\hat{\mathbf{w}})$. Therefore, $h(\mathbf{W}|\hat{\mathbf{W}})$ is equal to $h(\mathbf{Y}|\mathbf{X})$ for $\forall N \in \mathcal{N}$ for this distribution, since $h(\mathbf{Y}|\mathbf{X})$ does not depend on the distribution of $\mathbf{X}$. Furthermore, by marginalizing condition 4 in Definition 2.3, we get

$$p_{\mathbf{W}}(\mathbf{w}) = \int p_N(\mathbf{w}|\hat{\mathbf{w}})p_{\mathbf{W}}(\hat{\mathbf{w}})\, d\hat{\mathbf{w}},$$

which implies that if the channel input of the channel $p_N(\cdot|\cdot)$ follows $p_{\mathbf{W}}$, then the output also follows the same distribution $p_{\mathbf{W}}$. Therefore, condition 3 in Definition 2.2 is satisfied.

However, the converse is not true. Let us consider a source and channel pair with $\mathcal{X} = \mathcal{Y} = \mathcal{W} = \hat{\mathcal{W}} = \{0, 1, 2\}$. Let the source W be equiprobable on $\mathcal{W}$. Let $p_N(y|x) = 0$ if $y = (x+2) \bmod 3$ and $\frac{1}{2}$ otherwise. Let the distortion measure $d(w, \hat{w})$ for the source be 1 if $\hat{w} = (w + 2) \bmod 3$ and 0 otherwise. The expected distortion $D(N)$ of an uncoded system is always zero for this case. $H(Y)$ is maximized if the input X is equiprobable and thus property 3 of Definition 2.2 is satisfied. Since $D(N)$ is always zero, the joint distribution of W and $\hat{W}$ that maximizes $H(W|\hat{W})$

that gives zero distortion should be zero if $\hat{w} = (w+2) \bmod 3$. Therefore, $H(W|\hat{W})$ is upperbounded by one and it can be achieved by the following joint distribution of W and $\hat{W}$:

$$p^*(w,\hat{w}) = \begin{cases} 0, & \text{if } \hat{w} = (w+2) \bmod 3; \\ \frac{1}{6}, & \text{otherwise.} \end{cases}$$

Thus, this pair is entropy matched since property 4 of Definition 2.2 is also satisfied, where $H(Y|X) = 1$.

However, this pair is not density matched, since $p_W(\hat{w})p_N(w|\hat{w})$ is different from $p^*(w,\hat{w})$, i.e.,

$$p_W(\hat{w})p_N(w|\hat{w}) = \begin{cases} 0, & \text{if } \hat{w} = (w+1) \bmod 3; \\ \frac{1}{6}, & \text{otherwise.} \end{cases}$$

□

Many interesting matched sources and channels have additional structures. We define a group structure between the channel input and output as follows.

Definition 2.4 *A channel $p_N(\mathbf{y}|\mathbf{x})$ is an* additive-noise channel *if the following is satisfied:*

$$\mathbf{Y} = \mathbf{X} + \mathbf{Z},$$

where $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ are defined on an additive group $\langle \mathcal{X}, + \rangle$, $\mathbf{X}$ and $\mathbf{Z}$ are independent, and $\mathcal{X}$ is either a subset of $\mathbb{R}^m$ or $\mathbb{C}^m$, or a finite set.

If $\mathcal{X}$ is discrete, then an additive-noise channel is regular [15], and thus symmetric [28]. Gallager showed that the channel capacity of a symmetric channel can be achieved using a uniform input [28].

We will focus on two examples in which $\mathcal{X}$ is infinite. The first example is $\mathcal{X} = \mathbb{R}$ as in the AWGN channel. The second example is when $\mathcal{X}$ is a fundamental region $\mathcal{R}(\Lambda)$ of a lattice $\Lambda \in \mathbb{R}^m$, regarded as a quotient group $\mathbb{R}^m/\Lambda$.

If a channel has a group property, then it is easy to show that $h(\mathbf{Y}|\mathbf{X})$ depends only on $p(\mathbf{z})$ since $h(\mathbf{Y}|\mathbf{X}) = h(\mathbf{X}+\mathbf{Z}|\mathbf{X}) = h(\mathbf{Z}|\mathbf{X}) = h(\mathbf{Z})$. Furthermore, $p_N(\mathbf{y}|\mathbf{x})$ is equal to $p_{\mathbf{Z},N}(-\mathbf{x}+\mathbf{y})$ for this channel, where $-\mathbf{x}$ is the inverse element of $\mathbf{x}$ and $p_{\mathbf{Z},N}(\cdot)$ is the pdf of $\mathbf{Z}$ as a function of N. Abusing notation, we write $p_{\mathbf{Z},N}(\cdot)$ as $p_N(\cdot)$.

We define a difference-distortion measure as follows.

Definition 2.5 *A distortion measure $d(\mathbf{w}, \hat{\mathbf{w}})$ is a* difference-distortion *if*

$$d(\mathbf{w}, \hat{\mathbf{w}}) = \hat{d}(\mathbf{w} + (-\hat{\mathbf{w}})), \forall \mathbf{w}, \hat{\mathbf{w}} \in \mathcal{W},$$

where $\langle \mathcal{W}, + \rangle$ is an additive group and $\hat{d}(\cdot)$ is a mapping from $\mathcal{W}$ to $\mathbb{R}^+$.

We note that $\langle \mathcal{W}, + \rangle$ is different from $\langle \mathcal{X}, + \rangle$ in general. We assume $\mathcal{W}$ is either a subset of $\mathbb{R}^n$ or $\mathbb{C}^n$, or a finite set. If $\mathcal{W}$ is a fundamental region $\mathcal{R}(\Lambda)$ of a lattice $\Lambda \in \mathbb{R}^n$, regarded as a quotient group $\mathbb{R}^n/\Lambda$, then we define the difference distortion as the mod-Λ difference distortion (see Section 2.4).

2.3.1 Examples of Matched Sources and Channels

The following examples show matched sources and channels, in which each channel is given a different group property, and each source has a different difference-distortion measure.

Example 2.5 (Symmetric channel) *Assume a uniform source W with finite alphabet size n and the Hamming distortion measure $d(w, \hat{w}) = 1 - \delta_{w,\hat{w}}$. The rate-distortion function of this source is given by*

$$R(D) = \begin{cases} \log n - H(D) - D\log(n-1), & \text{if } 0 \le D \le (n-1)/n; \\ 0, & \text{otherwise,} \end{cases}$$

where $H(\cdot)$ is the binary entropy function. Now consider a symmetric channel with alphabet size n and error probability p, where a transition to a different symbol of the

alphabet occurs with probability $p/(n-1)$. The capacity of this channel is given by

$$C(p) = \log n - H(p) - p\log(n-1).$$

If $p \leq (n-1)/n$, then this source and channel are density matched.

Example 2.6 (BSC) *The rate-distortion function of a Bernoulli(1/2) source with a Hamming distortion measure is given by*

$$R(D) = \begin{cases} 1 - H(D), & \text{if } 0 \leq D \leq 1/2; \\ 0, & \text{otherwise.} \end{cases}$$

The channel capacity of a BSC with error probability p is given by

$$C(p) = 1 - H(p).$$

If $p \leq 1/2$, this source and channel are density matched. This is a special case of Example 2.5 when $n = 2$.

Example 2.7 (AWGN) *The rate-distortion function of a zero-mean variance-S Gaussian source $\sim \mathcal{N}(0, S)$ with a squared distortion measure is given by*

$$R(D) = \begin{cases} \frac{1}{2}\log\frac{S}{D}, & \text{if } 0 \leq D \leq S; \\ 0, & \text{otherwise.} \end{cases}$$

The channel capacity of an AWGN channel with input power constraint P and noise power N followed by a scalar gain of $\frac{P}{P+N}$ is given by

$$C(N) = \frac{1}{2}\log\left(1 + \frac{P}{N}\right).$$

If $P = S$, then this source and channel are matched, but not entropy matched.

The optimality of each uncoded system for the symmetric channel was shown in [38, 75], for the BSC was shown in [38, 14], and for the AWGN channel was shown in [30].

In Examples 2.5 and 2.6, the test channel for source coding is the same as the channel and the input and output distributions are the same.

We give another interesting example of a density-matched source and channel pair in the following three sections. First, we find the rate-distortion function of a uniform source and a difference-distortion measure in the following section. Then we show the channel capacity of a modulo channel with a group property (modulo addition) in Section 2.5. Finally, we show that this source and channel pair is density matched in Section 2.6.

2.4 A Uniform Source and Its Rate-Distortion Function

Wyner and Ziv [75] studied rate-distortion functions for bounded sources with various distortion measures. One of their examples was a uniform source on an interval, with a distortion measure that is periodic in the difference of two arguments, where one period is defined on the interval. In this section, we generalize this result to a uniform source defined on a fundamental region of an n-dimensional lattice.

An n-dimensional lattice Λ is defined as a discrete set of n-dimensional vectors that spans $\mathbb{R}^n$ and forms a group under ordinary vector addition in $\mathbb{R}^n$. A fundamental region $\mathcal{R}(\Lambda)$ of Λ is a subset of $\mathbb{R}^n$ whose translates by elements in Λ are disjoint and cover $\mathbb{R}^n$. The volume $V(\Lambda)$ of $\mathcal{R}(\Lambda)$ depends only on Λ and not on the shape of the fundamental region. The Voronoi region $\mathcal{R}_V(\Lambda)$ of Λ is defined as the set of points in the $\mathbb{R}^n$ that are at least as close to the origin $\mathbf{0}$ as to any other point in Λ; it is the closure of a fundamental region.

We define the uniform source on $\mathcal{R}(\Lambda)$ to be uniformly distributed on $\mathcal{R}(\Lambda)$. We define a mod-Λ difference-distortion measure, or simply a modulo-distortion measure, $d_\Lambda(\mathbf{w}, \hat{\mathbf{w}}) = d_\Lambda((\mathbf{w} - \hat{\mathbf{w}}) \bmod \Lambda)$ associated with $\mathcal{R}(\Lambda)$ as a mapping from $\mathbb{R}^n \times \mathbb{R}^n$ to $\mathbb{R}^+$, where a nonnegative function $d_\Lambda(\mathbf{w})$ is defined on $\mathcal{R}(\Lambda)$ and $(\mathbf{w} \bmod \Lambda)$ is defined to lie in $\mathcal{R}(\Lambda)$ for $\forall \mathbf{w} \in \mathbb{R}^n$. We define $\mathbf{x} = \mathbf{y} \bmod \Lambda$ if $\mathbf{x} - \mathbf{y} \in \Lambda$ for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$.

In general, $(\mathbf{w} - \hat{\mathbf{w}}) \bmod \Lambda$ and $(\mathbf{w} \bmod \Lambda) - (\hat{\mathbf{w}} \bmod \Lambda)$ are not equal. Note that $d_\Lambda(\mathbf{w}, \hat{\mathbf{w}})$ is not a metric in general. We do not impose any other condition on $d_\Lambda(\mathbf{w})$ except that we require it to be Lebesque-integrable on $\mathcal{R}(\Lambda)$ and bounded above and below.

Definition 2.6 *The* mod-Λ difference distortion $d_\Lambda(\mathbf{w}, \hat{\mathbf{w}})$ *between two vectors* $\mathbf{w}, \hat{\mathbf{w}} \in \mathbb{R}^n$ *is defined as*

$$d_\Lambda(\mathbf{w}, \hat{\mathbf{w}}) = d_\Lambda((\mathbf{w} - \hat{\mathbf{w}}) \bmod \Lambda),$$

where $d_\Lambda(\cdot)$ *is a real, nonnegative, bounded, Lebesque-integrable function on* $\mathcal{R}(\Lambda)$ *and* n *is the dimension of* $\mathcal{R}(\Lambda)$.

Note that this is a difference-distortion measure as defined in the previous section, where the associated group operation of two elements $\mathbf{x}, \mathbf{y} \in \mathcal{R}(\Lambda)$ is given by $(\mathbf{x} + \mathbf{y}) \bmod \Lambda$.

We define the r^{th}-power modulo-difference distortion measure as follows.

Definition 2.7 *The* r^{th}-power modulo-difference-distortion measure $d_\Lambda(\mathbf{w}, \hat{\mathbf{w}})$ *between two vectors* $\mathbf{w}, \hat{\mathbf{w}} \in \mathbb{R}^n$ *is defined as*

$$d_\Lambda(\mathbf{w}, \hat{\mathbf{w}}) = \|(\mathbf{w} - \hat{\mathbf{w}}) \bmod_V \Lambda\|^r,$$

where r *is a positive real number,* $\|\cdot\|$ *is the Euclidean norm, and we define* $\bmod_V$ *such that* $((\mathbf{w} - \hat{\mathbf{w}}) \bmod_V \Lambda) \in \mathcal{R}_V(\Lambda)$.[3] *If* $r = 2$, *then we will call* $d_\Lambda(\mathbf{w}, \hat{\mathbf{w}}) = \|(\mathbf{w} - \hat{\mathbf{w}}) \bmod_V \Lambda\|^2$ *a* modulo-squared-distortion *measure.*

Our analysis for uniform sources with a modulo-distortion measure will hold for an arbitrary fundamental region $\mathcal{R}(\Lambda)$ on which the source is defined. For example, the rate-distortion function for the source will remain unchanged even though different fundamental regions are used if $d_\Lambda(\cdot)$ is defined appropriately.

[3] *In the case of ambiguity on the boundary of* $\mathcal{R}_V(\Lambda)$, *we assume ties are broken arbitrarily. This does not affect the* r^{th}*-power modulo-difference distortion.*

For example, for the integer lattice $\mathbb{Z}$, we may choose the half-open Voronoi region $[-\frac{1}{2}, \frac{1}{2})$ for $\mathcal{R}(\mathbb{Z})$ and choose $d_{\mathbb{Z}}(x) = x^2$, where $x \in \mathcal{R}(\mathbb{Z})$. We may also choose a fundamental region $\mathcal{R}(\mathbb{Z}) = [0, 1)$ and choose $d_{\mathbb{Z}}(x) = x^2$ if $x < \frac{1}{2}$ and $d_{\mathbb{Z}}(x) = (x-1)^2$ if $x \geq \frac{1}{2}$. Even though we have two different definitions for $\mathcal{R}(\mathbb{Z})$ and $d_{\mathbb{Z}}(\cdot)$, the modulo distortion $d_{\mathbb{Z}}(\mathbf{w}, \hat{\mathbf{w}}), \mathbf{w}, \hat{\mathbf{w}} \in \mathbb{R}$ is the same.

In general, the support of a uniform source can be always mapped to a fundamental region $\mathcal{R}(\Lambda)$ by using a mod-Λ mapping without affecting the modulo distortion. The reason we consider modulo-distortion measures in Chapters 2 and 3 is because they introduce symmetry and make analysis cleaner. For example, it makes optimal distribution of reconstruction points uniform.

For high-rate quantization, a modulo-squared-distortion measure such as $||(\mathbf{w} - \hat{\mathbf{w}}) \bmod_V \Lambda||^2$ becomes equivalent to the usual squared-error distortion $||\mathbf{w}-\hat{\mathbf{w}}||^2$ with a high probability, since the probability that the error vector $\mathbf{w}-\hat{\mathbf{w}}$ lies outside $\mathcal{R}_V(\Lambda)$ approaches zero as the rate tends to infinity. Therefore, for high-rate quantization, we may use modulo distortion as an approximation to the usual difference-distortion measure. Practically, the difference between two distortion measures is negligible unless the SNR is very small.

The following truncated exponential density function will be useful in our analysis.

Definition 2.8 *The* truncated exponential density function $f_{d_\Lambda, D}(\cdot)$ *associated with* $d_\Lambda(\cdot)$ *is a density function such that*

$$f_{d_\Lambda, D}(\mathbf{x}) = \begin{cases} c\, e^{-\mu d_\Lambda(\mathbf{x})}, & \text{if } \mathbf{x} \in \mathcal{R}(\Lambda); \\ 0, & \text{otherwise,} \end{cases} \tag{2.2}$$

where $D \in \mathbb{R}$ *is given and* $c, \mu \in \mathbb{R}$ *satisfy*

$$\int_{\mathcal{R}(\Lambda)} f_{d_\Lambda, D}(\mathbf{x})\, d\mathbf{x} = 1$$

and

$$\int_{\mathcal{R}(\Lambda)} d_\Lambda(\mathbf{x}) f_{d_\Lambda, D}(\mathbf{x})\, d\mathbf{x} = D.$$

When $d_\Lambda(\cdot)$ is a modulo-squared-distortion measure, then we call $f_{d_\Lambda,D}(\cdot)$ the *truncated Gaussian* density function.

Note that c and D are uniquely determined if μ is given. The differential entropy of $f_{d_\Lambda,D}$ in nats can be written using D, c and μ as follows:

$$\begin{aligned} h(f_{d_\Lambda,D}) &= -\int_{\mathcal{R}(\Lambda)} f_{d_\Lambda,D}(\mathbf{x}) \log f_{d_\Lambda,D}(\mathbf{x})\, d\mathbf{x} \\ &= -\int_{\mathcal{R}(\Lambda)} f_{d_\Lambda,D}(\mathbf{x}) \left[\log c - \mu d_\Lambda(\mathbf{x})\right]\, d\mathbf{x} \\ &= -\log c + \mu D. \end{aligned}$$

It is not trivial from the above derivation, but the differential entropy is unique for a given D, as the following lemma shows. See Appendix A for a proof.

Lemma 2.4 *The differential entropy $h(f_{d_\Lambda,D})$ of a truncated exponential density function $f_{d_\Lambda,D}$ increases monotonically as D increases if $\mu > 0$, and decreases monotonically as D increases if $\mu < 0$. Furthermore, it is uniquely determined for a given D.*

Note that $D(\mu)$ is monotonically non-increasing as μ increases. It is clear from the lemma that the differential entropy of $f_{d_\Lambda,D}$ is maximized if $\mu = 0$, corresponding to the well-known fact that the uniform distribution maximizes the differential entropy when its support is finite. Therefore, the differential entropy of $f_{d_\Lambda,D}$ is upperbounded by $\log V(\Lambda)$, which is achievable if $\mu = 0$ or equivalently if $D = D_0$, where we define $D_0 = D(0)$. If $D < D_0$, then D can be considered a measure of uncertainty of the random variable, because it behaves like entropy; i.e., D increases/decreases monotonically as the entropy increases/decreases.

The following lemma shows that the truncated exponential density has a maximum-entropy property. A proof is given in Appendix A.

Lemma 2.5 *Let the random vectors $\mathbf{X}, \mathbf{Y} \in \mathcal{R}(\Lambda) \subset \mathbb{R}^n$ have the expected distortion D, i.e., $E[d_\Lambda(\mathbf{X})] = D$ and $E[d_\Lambda(\mathbf{Y})] = D$. Let $\mathbf{Y}$ have the truncated exponential density function associated with $d_\Lambda(\cdot)$. Then $h(\mathbf{X}) \le h(\mathbf{Y})$, with equality iff $\mathbf{X} = \mathbf{Y}$ almost everywhere.*

Let us assume that we have a length-k source that generates an iid sequence of vectors $\mathbf{W}_1^k = \{\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_k\}$, where each vector $\mathbf{W}_i, 1 \le i \le k$ is uniform in $\mathcal{R}(\Lambda)$. We want to design a codebook $\mathcal{C}$ to quantize this source, where the codebook has M codewords $\{\hat{\mathbf{W}}_{1,1}^k, \hat{\mathbf{W}}_{1,2}^k, \ldots, \hat{\mathbf{W}}_{1,M}^k\}$ and the j-th codeword $\hat{\mathbf{W}}_{1,j}^k, 1 \le j \le M$, is a sequence of vectors in the k-fold Cartesian product $\mathcal{R}(\Lambda)^k$ of a fundamental region $\mathcal{R}(\Lambda)$ of the lattice Λ. Note that the product space $\mathcal{R}(\Lambda)^k$ has a dimension of nk, where n is the dimension of $\mathcal{R}(\Lambda)$. The following theorem shows how to calculate the rate-distortion function $R(D)$ for an iid uniform source $\mathbf{W}$ in $\mathcal{R}(\Lambda)$ with a modulo-difference-distortion measure $d_\Lambda(\cdot,\cdot)$. See Appendix A for a proof.

Theorem 2.6 *The rate-distortion function $R(D)$ of an iid uniform source $\mathbf{W}$ in $\mathcal{R}(\Lambda)$ with a modulo-difference distortion $d_\Lambda(\cdot,\cdot)$ is*

$$R(D) = \begin{cases} \log V(\Lambda) - h(f_{d_\Lambda,D}), & \text{if } D \le D_0; \\ 0, & \text{otherwise.} \end{cases} \tag{2.3}$$

As an example, we take $\Lambda = \mathbb{Z}$, a fundamental region $\mathcal{R}(\mathbb{Z}) = [-\frac{1}{2}, \frac{1}{2})$, and a uniform source W defined on $\mathcal{R}(\mathbb{Z})$. Given this source distribution, we may have several different definitions of distortion, each of which will result in a different rate-distortion function in general. For example, we may take a modulo-squared-distortion measure $d_\mathbb{Z}(w, \hat{w}) = |(w - \hat{w}) \bmod_V \mathbb{Z}|^2$. Note that this distortion is the same as the regular MSE distortion if $|w - \hat{w}| \le \frac{1}{2}$. Therefore, the rate-distortion function using the modulo distortion is a good approximation to the rate-distortion function using the MSE distortion when the distortion is small. Moreover, if we think of the interval $\mathcal{R}(\mathbb{Z})$ as a circle S^1, then the source x becomes a uniformly distributed phase signal and the modulo distortion can be interpreted as the squared phase error [75]. The absolute phase error is $|(w - \hat{w}) \bmod_V \mathbb{Z}|$.

The rate-distortion function $R(D)$ of W with the modulo distortion $d_\mathbb{Z}(\cdot,\cdot)$ is given

by

$$R(D) = \begin{cases} -h(g_D) = \log c - \mu D, & \text{if } D \le D_0; \\ 0, & \text{otherwise,} \end{cases}$$

where $D_0 = D(\mu = 0) = \frac{1}{12}$, c and μ are functions of D, and g_D is the truncated Gaussian density given by

$$g_D(z) = \begin{cases} ce^{-\mu d_{\mathbb{Z}}(z)}, & \text{if } z \in \mathcal{R}(\mathbb{Z}); \\ 0, & \text{otherwise,} \end{cases}$$

where c and μ satisfy $E[d_{\mathbb{Z}}(Z)] = D$ and the normalization condition. The test channel for this case consists of an iid uniform input $\hat{W}$ followed by an additive noise $Z \sim g_D(z)$ followed by a mod-$\mathbb{Z}$ mapping.

2.5 A Modulo-Additive Noise Channel and Its Capacity

The modulo-additive white noise (MAWN) channel is defined as an additive iid noise followed by a modulo mapping mod-Λ, where Λ is an n-dimensional lattice. We will assume that the output of the modulo mapping is in a fundamental region $\mathcal{R}(\Lambda)$ of Λ. In general, this channel lattice is different from the source lattice in the previous section.

MAWN channels, or mod-Λ channels were used in [17, 22]. In this section, we summarize the mod-Λ channel capacity results in [21, 22].

The input, output, and noise are all n-dimensional vectors. Let us denote the channel input by $\mathbf{X}$, the output by $\mathbf{Y}$, and the noise by $\mathbf{Z}$. Then,

$$\mathbf{Y} = (\mathbf{X} + \mathbf{Z}) \bmod \Lambda.$$

The output $\mathbf{Y}$ has finite support $\mathcal{R}(\Lambda)$. Therefore, if the support of the noise $\mathbf{Z}$

includes $\mathcal{R}(\Lambda)$, then we can use $\mathbf{Z}' = (\mathbf{Z} \bmod \Lambda)$ instead of $\mathbf{Z}$, since this does not change the output $\mathbf{Y}$.

The pdf of $\mathbf{Z}' = (\mathbf{Z} \bmod \Lambda)$ becomes the aliased density of $p_{\mathbf{Z}}$, i.e.,

$$p_{\mathbf{Z}'}(\mathbf{z}') = \begin{cases} \sum_{\mathbf{x}\in\Lambda} p_{\mathbf{Z}}(\mathbf{z}' - \mathbf{x}), & \text{if } \mathbf{z}' \in \mathcal{R}(\Lambda); \\ 0, & \text{otherwise,} \end{cases}$$

where $p_{\mathbf{Z}'}$ is the pdf of $\mathbf{Z}'$ and $p_{\mathbf{Z}}$ is the pdf of $\mathbf{Z}$. Therefore, without loss of generality, we assume that the noise $\mathbf{Z}$ has finite support $\mathcal{R}(\Lambda)$.

We define the noise power N of the noise $\mathbf{Z} \in \mathcal{R}(\Lambda)$ as the expected value of $r(\mathbf{Z})$, where $r(\mathbf{Z})$ is a mapping from $\mathcal{R}(\Lambda)$ to $\mathbb{R}^+$. For example, $r(\mathbf{Z}) = ||\mathbf{Z}||^2$ can be used to define the noise power in the usual sense.

The channel capacity C of the MAWN channel as a function of N is defined as

$$\begin{aligned} C(N) &= \max_{p(\mathbf{x})} I(\mathbf{X};\mathbf{Y}) \\ &= \max_{p(\mathbf{x})} h(\mathbf{Y}) - h(\mathbf{Y}|\mathbf{X}), \end{aligned}$$

where $h(\mathbf{Y})$ is the differential entropy of $\mathbf{Y}$, $h(\mathbf{Y}|\mathbf{X})$ is the conditional differential entropy of $\mathbf{Y}$ given $\mathbf{X}$, and $p(\mathbf{x})$ is the pdf of $\mathbf{X}$.

As shown in [21, 22], since this channel is symmetric, the optimal input distribution is uniform, i.e., $p(\mathbf{x}) = 1/V(\Lambda)$, $\mathbf{x} \in \mathcal{R}(\Lambda)$. The output distribution is also uniform by the symmetry of the channel. Therefore,

$$\begin{aligned} h(\mathbf{Y}) &= -\int_{\mathcal{R}(\Lambda)} p(\mathbf{y}) \log p(\mathbf{y})\, d\mathbf{y} \\ &= \log V(\Lambda) \end{aligned}$$

and

$$\begin{aligned} h(\mathbf{Y}|\mathbf{X}) &= E_{\mathbf{x}}[h(\mathbf{Y}|\mathbf{X} = \mathbf{x})] \\ &= h(\mathbf{Y}|\mathbf{X} = 0) \\ &= h(\mathbf{Z}), \end{aligned}$$

where we used the fact that $h(\mathbf{Y}|\mathbf{X} = \mathbf{x})$ is independent of $\mathbf{x}$ due to symmetry. Therefore, the channel capacity is

$$C(N) = \log V(\Lambda) - h(\mathbf{Z}). \tag{2.4}$$

2.6 Optimal Joint Coding for the Uniform Source and the MAWN Channel

In this section, we show that under certain assumptions a simple uncoded system to transmit an iid uniform source through an MAWN channel is optimal for all values of SNR. Here we assume the uniform iid source of Section 2.4 and the MAWN channel of Section 2.5. We further assume that the source and the channel are based on the same n-dimensional lattice Λ; i.e., the source is uniform on a fundamental region $\mathcal{R}(\Lambda)$, a modulo-distortion measure $d_\Lambda(\cdot)$ is defined on $\mathcal{R}(\Lambda)$, and the channel input, output, and noise are in $\mathcal{R}(\Lambda)$.

We first assume that the noise power N of the noise $\mathbf{Z}$ is given by the expected value of $d_\Lambda(\mathbf{Z})$, where d_Λ is the modulo-distortion measure for the source. We also assume that the noise density $p_N(\mathbf{z})$ of $\mathbf{Z}$ is the truncated exponential density $f_{d_\Lambda,N}(\mathbf{z})$, i.e.,

$$f_{d_\Lambda,N}(\mathbf{z}) = \begin{cases} c\, e^{-\mu d_\Lambda(\mathbf{z})}, & \text{if } \mathbf{z} \in \mathcal{R}(\Lambda); \\ 0, & \text{otherwise,} \end{cases}$$

where c, μ are functions of N.

Since the rate-distortion function $R(D)$ of the source $\mathbf{W}$ is given by (2.3) and the channel capacity of the MAWN channel is given by (2.4), the test channel for the source becomes precisely the MAWN channel under the given assumptions, which implies that this source and channel pair satisfies condition 4 of Definition 2.3. Since this source and channel pair also satisfies the other three conditions of Definition 2.3, it is density matched.

Therefore, if we send the uncoded signal $\mathbf{W}$ through the channel, then the distortion $D(N)$ we get is the same as the noise power N and this scheme achieves the Shannon limit for all values of N.

For example, for a uniform source on $\mathcal{R}(\mathbb{Z}) = [-\frac{1}{2}, \frac{1}{2})$ and the distortion measure $d_{\mathbb{Z}}(w) = |w|^2, w \in \mathcal{R}(\mathbb{Z})$, an uncoded system achieves the Shannon limit for all values of SNR if the channel is a modulo-additive white truncated-Gaussian noise channel.

2.7 Summary

In this chapter, we have defined matched sources and channels and have shown some examples. We have also constructed the rate-distortion function of a uniform source over a fundamental region of a lattice with a modulo-difference-distortion measure. We have shown the optimality of an uncoded system for the uniform source and a modulo-white additive noise channel.

Chapter 3

Joint Source-Channel Coding Using Space-Filling Curves

In this chapter, we consider joint source-channel coding schemes using curves when the source bandwidth is greater than the channel bandwidth. We will construct various joint coding schemes for both uniform and Gaussian sources using space-filling curves and other families of curves. We will use the same notations as in the previous chapter.

3.1 Introduction

3.1.1 Bandwidth Expansion Using Curves

Suppose we have a one-dimensional analog source with a squared-distortion measure and an m-dimensional analog channel, where $m > 1$. Then the mapping from the source to the channel input is a curve (discontinuous in general).

This type of coding scheme is used in practice, for instance, in frequency modulation (FM) and pulse position modulation (PPM). Shannon described some examples of such mappings in one of his original papers [60]. One of his examples is shown in Figure 3-1, where a one-dimensional source is mapped to a two-dimensional plane via a curve.

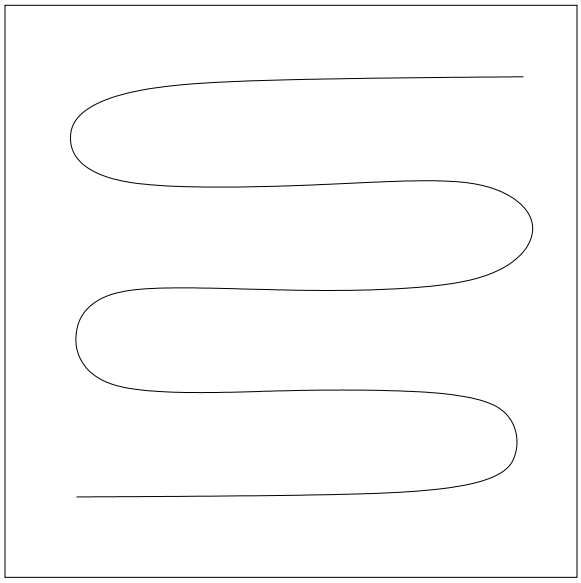


Figure 3-1: Bandwidth Expansion Using a Two-Dimensional Curve

When the noise is small, the distortion will be very small if the curve is stretched long enough. However, if the noise level exceeds a certain threshold, then a large distortion will occur due to a discontinuous jump. Therefore, this system could transmit analog signals with less distortion than linear modulation schemes by increasing the bandwidth. Wozencraft and Jacobs did more rigorous analysis on this type of "twisted modulation" scheme [74].

In this thesis, however, we will focus on bandwidth reduction schemes for reasons described in the next section.

3.1.2 Bandwidth Reduction Using Curves

In modern digital transmission systems the greater concern is efficient use of bandwidth. For example, by compressing an NTSC video signal, we can send about six digital video channels in one 6 MHz band, equal to the bandwidth of one analog NTSC channel.

In this chapter, we show how to use a curve to construct a bandwidth-reduction scheme. Our goal is to construct joint coding schemes that show graceful degradation when the channel is mismatched.

We require such curves be continuous due to the following reason. Suppose a discontinuous curve is used to map an interval $I = [0, 1]$ to a square I^2 as in the following Cantor-type example by Shannon [60]:[1]

$$\left.\begin{array}{l} x = .a_1a_2a_3\ldots \\ y = .b_1b_2b_3\ldots \end{array}\right\} \Leftrightarrow z = .a_1b_1a_2b_2a_3b_3\ldots \ ,$$

where x, y, and z are represented in binary numbers and each digit in z is taken from x and y alternatively. However, since the curve defined by the above mapping is discontinuous, the position (x, y) in I^2 will change dramatically in the presence of a small noise in the interval I. In the next section, we will show how to make this curve continuous by using space-filling curves.

We will consider two cases of bandwidth-reduction schemes in this chapter. The first case is when an n-dimensional source $\mathbf{W}$ is uniform on a fundamental region $\mathcal{R}(\Lambda_s)$ of the source lattice Λ_s with a modulo-difference-distortion measure $d_{\Lambda_s}(\cdot,\cdot)$ and when the channel is a MAWN channel with a truncated exponential noise density $f_{d_{\Lambda_c},D}(\cdot)$ defined on $\mathcal{R}(\Lambda_c)$ of a channel lattice Λ_c as shown in (2.2). As shown in the previous chapter, if the source and the channel lattices are the same, this source and channel pair is matched and therefore an uncoded system achieves the Shannon limit for all values of the channel SNR.

In this chapter, we will consider cases when the source lattice Λ_s is $\mathbb{Z}^2$, $\mathbb{Z}^3$, or A_2 and the channel lattice Λ_c is $\mathbb{Z}$, which reduces the source bandwidth by 2:1 or 3:1. We assume the source is uniform on $\mathbf{W} = \mathcal{R}(\Lambda_s)$, where $\mathcal{R}(\Lambda_s)$ is a fundamental region of the source lattice. We also assume a modulo-squared-distortion measure is defined on the Voronoi region $\mathcal{R}_V(\Lambda_s)$ of Λ_s. We assume the channel noise has a truncated-Gaussian density defined on $\mathcal{R}_V(\mathbb{Z})$.

The second case is when the source is two-dimensional Gaussian with iid components and when the channel is AWGN with an input power constraint. We assume squared distortion for the source. In this case, we obtain a 2:1 bandwidth reduction.

[1]This was just an example of a bandwidth reduction scheme and it seems that Shannon did not intend to build a joint source-channel scheme based on this example.

We define an n-dimensional curve $\mathbf{c}(t)$ as a mapping from $\mathcal{T} \subset \mathcal{X} \subset \mathbb{R}$ to $\mathcal{W} \subset \mathbb{R}^n$, where $\mathcal{X}$ is the channel input space and n is the dimension of the source.

The encoder tries to find a point $\mathbf{c}(\hat{t})$ on the curve $\mathbf{c}(t)$ that is closest to the source point $\mathbf{w} \in \mathcal{W}$; i.e.,

$$\hat{t}(\mathbf{w}) = \underset{t \in \mathcal{T}}{\operatorname{argmin}} ||\mathbf{w} - \mathbf{c}(t)||.$$

For the Gaussian case, the optimal decoder $\hat{g}(\cdot)$ given an encoder is an MMSE estimator; i.e.,

$$\hat{g}(y) = E[\mathbf{W}|Y = y],$$

where Y is the channel output. For the uniform case, MMSE estimation becomes a minimum modulo-mean-square error estimation problem. We will show how to solve this problem in Section 3.3.

The optimal curve $\hat{\mathbf{c}}(t)$, if exists, needs to have a property such that, when a small noise is added to t, $\hat{\mathbf{c}}(t)$ is not changed very much. In other words, it needs to have a *locality-preserving* property.

We will use space-filling curves and other families of curves to design $\mathbf{c}(t)$ in Sections 3.3 and 3.4.

3.1.3 Bandwidth Reduction Using Numerically Optimized Vector Quantizers

For the Gaussian case, there have been some prior works on designing joint source-channel coding schemes using vector quantizers (VQs). The key design problems are (1) designing a source quantizer, (2) finding a mapping from the output index of the source quantizer to the input index of the channel modulator, and (3) designing the channel modulator.

Some numerical algorithms were developed in [18, 65] to numerically design a vector quantizer to minimize the overall distortion in the presence of the channel

noise. This was generalized to the case when the channel input is power constrained in [25, 24], which is called *power-constrained channel-optimized vector quantization* (PCCOVQ). Their algorithm was developed by modifying the generalized Lloyd algorithm to take into account the channel noise and the power constraint. However, due to the complexity and the stability of the algorithm, the codebook size is limited.

In Section 3.4, we will provide analysis of some of the results in [24] for Gaussian channels and show it is possible to find optimal coding schemes analytically when the coding scheme is restricted to spiral or spiral-like curves.

In this section, we show some examples of vector quantizers optimized using PCCOVQ for two cases described in the previous section, i.e., (1) uniform source[2] and the mod-$\mathbb{Z}$ channel with truncated Gaussian noise[3] and (2) the Gaussian source and the AWGN channel. We refer readers to [24] for details of this algorithm.

In Sections 3.3 and 3.4, we compare these vector quantizers with our joint source-channel coding schemes designed using space-filling curves and other families of curves. We show that our coding schemes are slightly better than the numerically optimized schemes in this section.

Figure 3-2 shows four 256-point vector quantizers designed at channel SNR = 11.3, 19.3, 33.7, and 51.6 dB, respectively, when the source is iid uniform on I, the channel is mod-$\mathbb{Z}$, and the noise is truncated Gaussian. A bandwidth reduction of 2:1 is used.

As we increase the target SNR of the vector quantizer, we can observe finer structures. Triangular structures do not appear until the target SNR is about 48 dB.

Figure 3-3 shows SDRs as functions of the channel SNR using the four 256-point vector quantizers in Figure 3-2. We observe that the performance is quite good for the quantizer designed for the channel SNR = 33.7 dB. The performance of the quantizer designed for the channel SNR = 51.6 dB shows flatness at about SDR = 24 dB, which is due to the fact that there is only a finite number of quantization points.

[2]We have used a squared-error distortion measure instead of modulo distortion in this section for simplicity. However, the difference in two performances at mid to high SNRs should be negligible.

[3]There is no power constraint in this case and the problem of designing a channel-optimized vector quantizer becomes slightly simpler.

Table 3.1: Peak performance in SDR gap in dB from the Shannon limit for (a) uniform and (b) Gaussian cases, where a 2:1 B/W reduction is achieved using numerically optimized vector quantizers.

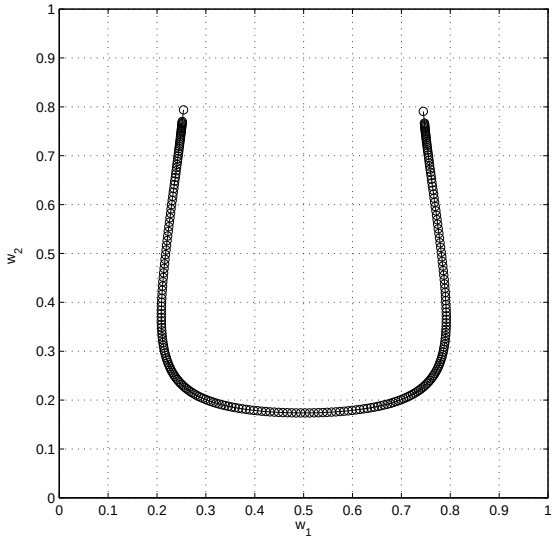
Uniform			Gaussian		
target SNR	best	at [dB]	target SNR	best	at [dB]
11.3			2.3	1.2	6
19.3	1.9	16	13.0	1.3	16
33.7	1.3	33	25.2	1.2	25
51.6	2.5	50	47.8	3.0	45

Figure 3-4 shows four 256-point vector quantizers designed at channel SNR = 2.3, 13.0, 25.2, 47.8 dB, respectively, when the source is iid Gaussian with unit variance and the channel is AWGN, using the algorithm developed in [24]. A bandwidth reduction of 2:1 is used. This confirms the results in [24], where similar parameters were used.

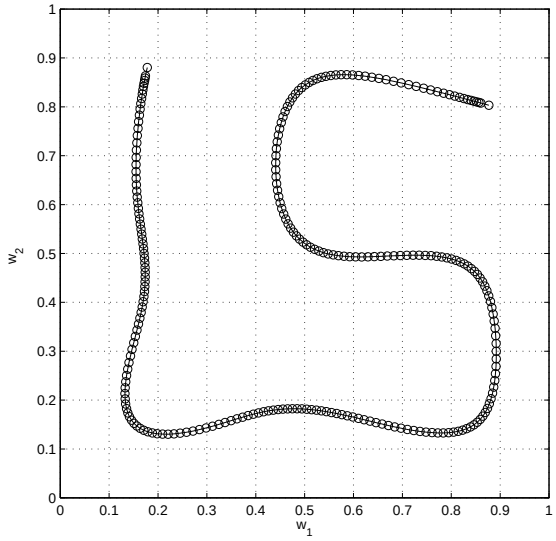
An interesting observation here is that, at mid or low SNRs, the 256-point vector quantizer becomes spiral-like, but the constellation becomes complicated if the target SNR is very high. Triangular structures begin to appear when the target SNR is about 43 dB.

Figure 3-5 shows SDRs as functions of the channel SNR using the four vector quantizers in Figure 3-4. We observe that the peak performance is quite good for quantizers designed at mid to low SNRs (spiral shaped), but is not for the quantizer designed for SNR = 47.8 dB.

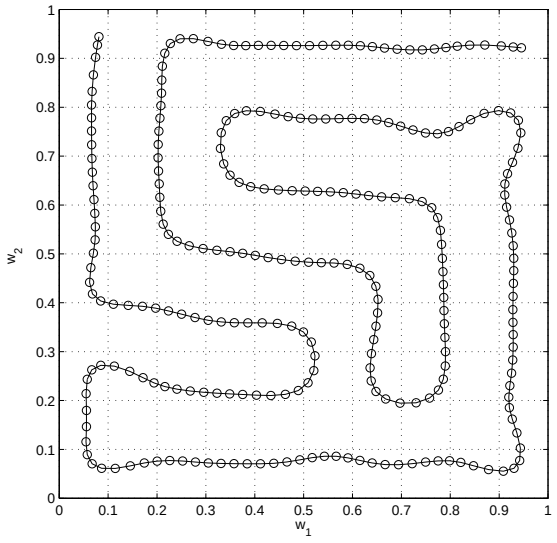
Table 3.1 shows the peak performance of each coding scheme given in this section. For the uniform case, we can achieve about 1.3 dB from the Shannon limit and for the Gaussian case, we can achieve about 1.2 dB from the Shannon limit.



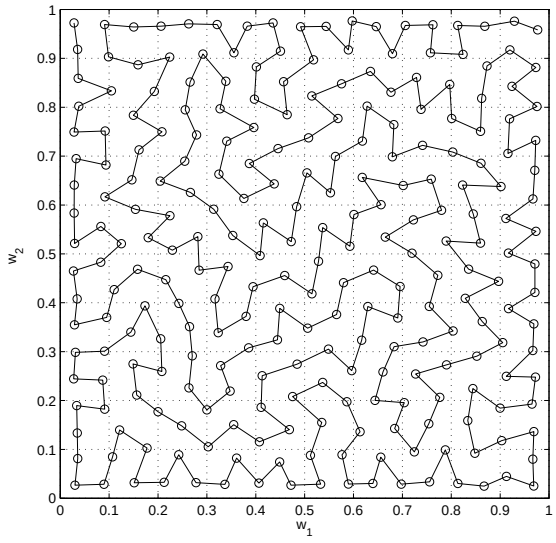
(a) Target SNR = 11.3 dB



(b) Target SNR = 19.3 dB



(c) Target SNR = 33.7 dB



(d) Target SNR = 57.6 dB

Figure 3-2: 256-point vector quantizers optimized for the uniform case

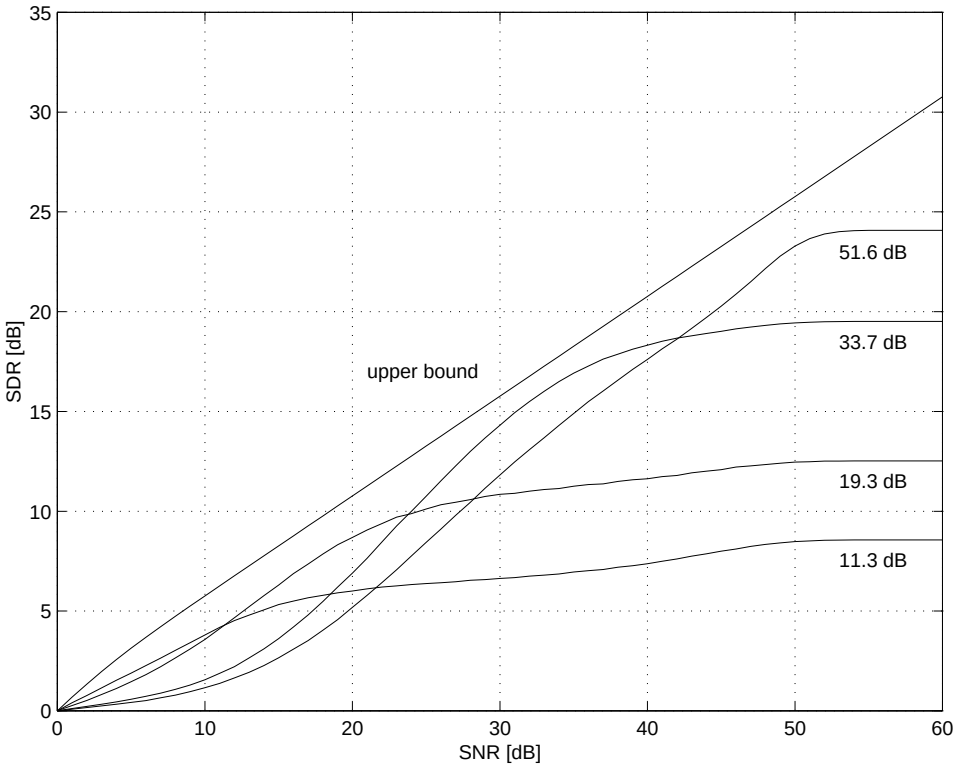
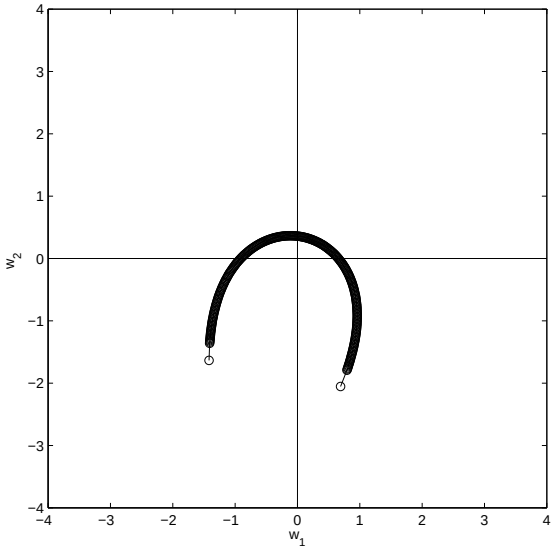
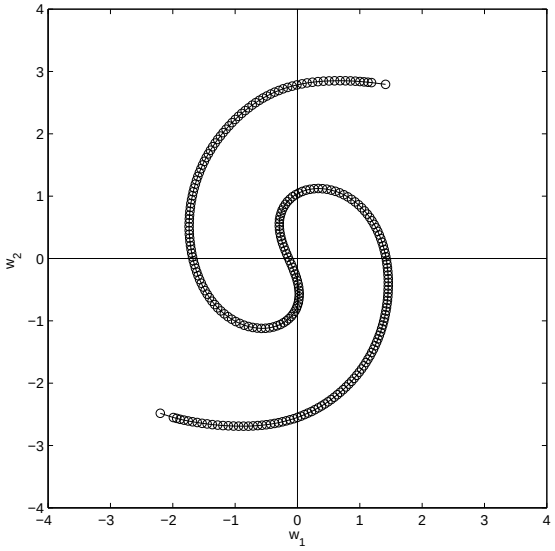


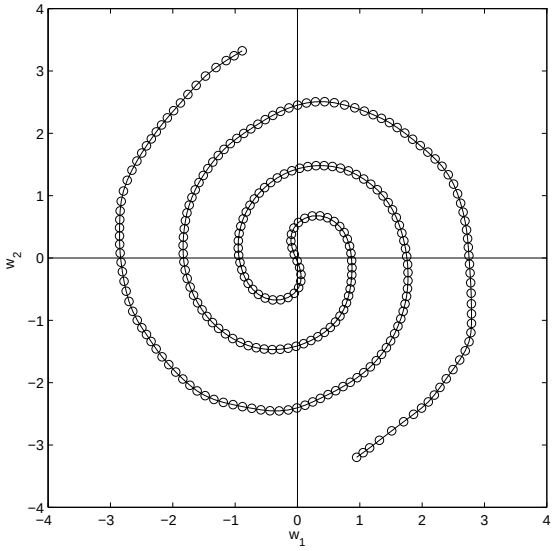
Figure 3-3: SDR vs. SNR plots for the four 256-point vector quantizers in Figure 3-2 for the uniform case. The top solid line is the Shannon limit.



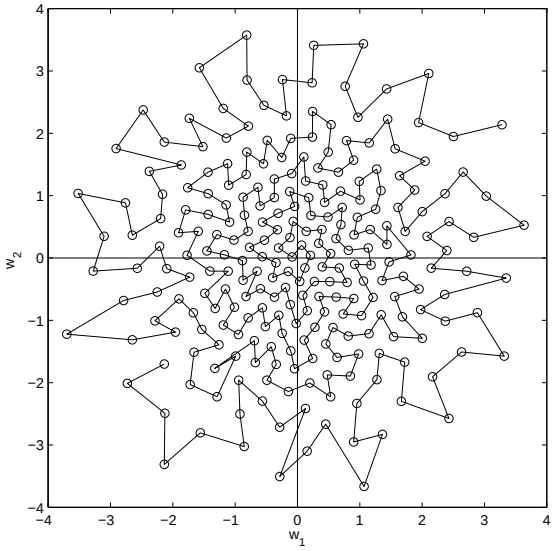
(a) Target SNR = 2.3 dB



(b) Target SNR = 13.0 dB



(c) Target SNR = 25.2 dB



(d) Target SNR = 55.3 dB

Figure 3-4: 256-point vector quantizers for the Gaussian case. The numerical optimization algorithm (PCCOVQ) of Fuldseth [24] was used.

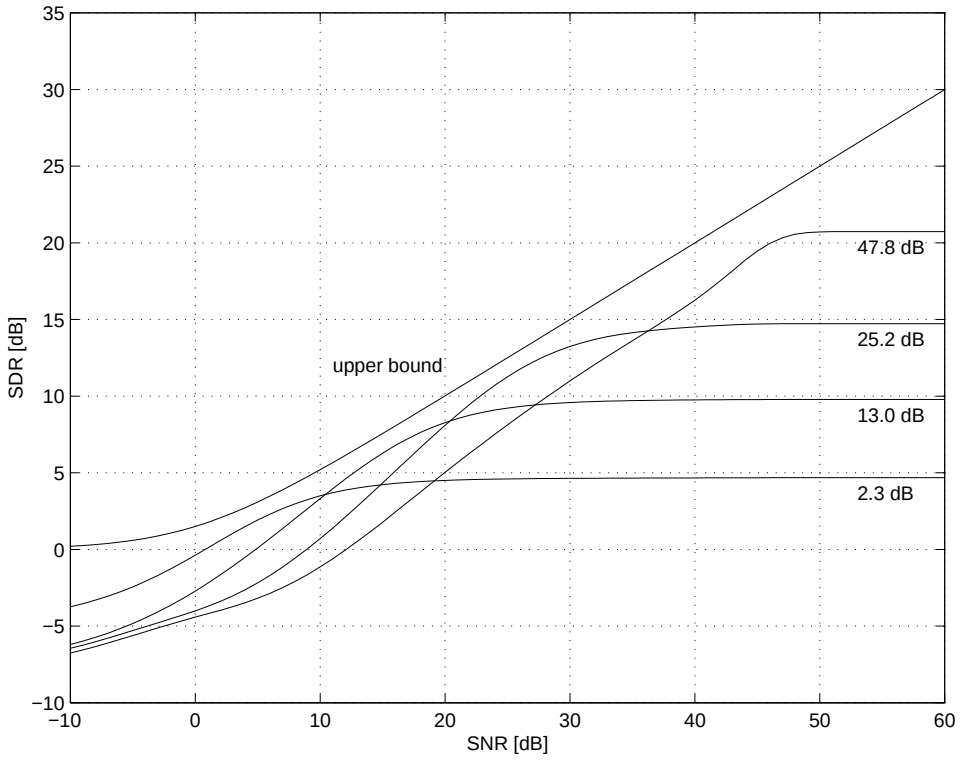


Figure 3-5: SDR vs. SNR plots for the four 256-point vector quantizers in Figure 3-4 for the Gaussian case. The top solid line is the Shannon limit.

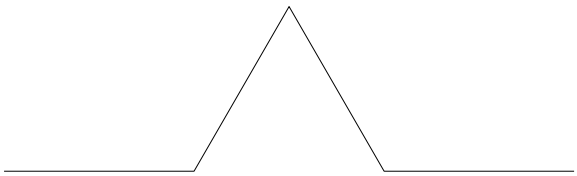
3.2 Space-Filling Curves

The term *fractals* describe mathematical objects having *self-similarity* on all scales. An interesting fact about a fractal is that it can have a fractional dimension, which is called *the fractal dimension*, *the similarity dimension*, or the *Hausdorff-Besicovitch dimension*. The fractal dimension $d = \log_s N$ of a set of length L is defined such that when the set can be divided into N self-similar subsets of length L/s [47]. For example, an n-cube has the fractal dimension n. Formally, a fractal is defined as an object whose fractal dimension is strictly greater than its topological dimension [47]. Therefore, an n-cube is not a fractal because its topological dimension n is equal to its fractal dimension n.

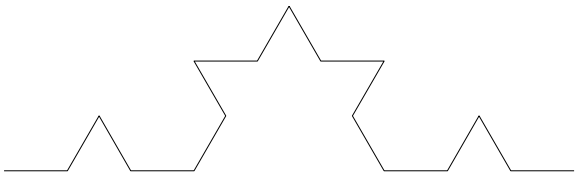
As a special case of fractals, we can generate a fractal curve using a string-rewriting system, also called an L-system or a Lindenmeyer-system. An L-system is generated by first beginning with an *axiom* and then applying *production rules* recursively. For example, starting with an axiom `F` and using a rewriting rule `F` $\rightarrow$ `F+F--F+F`, we get `F+F--F+F` after one iteration, which may be interpreted as a string of commands for a small robot on a plane, i.e., `F` as going forward one unit, `+` as a counterclockwise rotation of a unit angle, and `-` as a clockwise rotation of a unit angle. Applying this rule again, we get `F+F--F+F+F+F--F+F--F+F--F+F+F+F--F+F`, and we can keep doing this iteration while maintaining the the end-to-end distance constant as shown in Figure 3-6. (We assume that the unit angle is $\frac{\pi}{3}$ for this example.) The limit of the sequence of the curves of this example is known as the Koch island, which has the fractal dimension $\log_3 4 \sim 1.26$.

We now define a space-filling curve as a continuous surjective mapping from a closed interval $I = [0, 1] \in \mathbb{R}$ onto an n-dimensional cube $I^n = [0, 1]^n$, where $n > 1$.[4] In other words, this curve passes through every point in the n-dimensional cube.

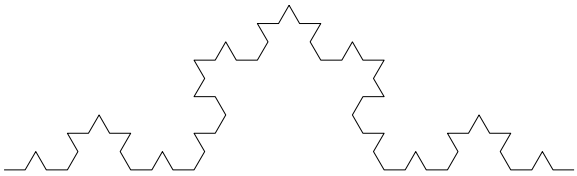
[4] A bijective mapping from I to I^n, where $n > 1$, is necessarily discontinuous [58]. A Jordan curve is a continuous injective mapping from I to $\mathbb{E}^n$, where $\mathbb{E}^n$ is an n-dimensional Euclidean space. Jordan curves in $\mathbb{E}^n$ with positive n-dimensional Lebesque measure are called Osgood curves [58]. In general, a continuous surjective mapping from I to a Hausdorff space X exists if and only if X is compact, connected, locally connected, and metrizable, which is the famous Hahn-Mazurkiewicz theorem.



(a) After 1 iteration (`F+F--F+F`)



(b) After 2 iterations
(`F+F--F+F+F+F--F+F--F+F--F+F+F+F--F+F`)



(c) After 3 iterations
(`F+F--F+F+F+F--F+F--F+` ... `+F--F+F--F+F+F+F--F+F`)

Figure 3-6: Generation of the Koch island

Curves with this property are called Peano's space-filling curves or simply space-filling curves.[5]

Peano [49] constructed the first such curve, which can be generated by an L-system with an axiom `L` and a set of rules `L` → `LFRFL-F-RFLFR+F+LFRFL` and `R` → `RFLFR+F+LFRFL-F-RFLFR` as shown in Figure 3-7 (a). The limit of the sequence is the Peano curve, which satisfies the continuity and surjectivity conditions. Another surprising fact about the Peano curve is that its coordinate functions are nowhere differentiable [58].

Two more versions of the Peano curve are shown in Figure 3-7 (b) and (c).[6] The switch-back type (b) can be generated by an L-system with an axiom `L` and a rule `L` → `L+FLF-LF-LF-L+FL+FL+FLF-L`. The meander type (c) can be generated by an L-system with an axiom `L` and a set of rules `L` → `LF-RFR-FL+F+LFLFL+FRFR-` and `R` → `RF+LFL+FR-F-RF RFR-FLFL+`.

Another famous example is the Hilbert curve [36], which can be defined by an L-system with an axiom `L` and a set of rules `L` → `+RF-LFL-FR+` and `R` → `-LF+RFR+FL-` as illustrated in Figure 3-8 (a), where the Hilbert curve is the limit of the sequence. The Hilbert curve also satisfies the continuity and surjectivity, and its coordinate functions are nowhere differentiable.

Figure 3-8 (b) shows Moor's version of Hilbert's space-filling curve [58]. It can be generated by an L-system with an axiom `S` and a set of rules `S` → `+LFL-F-LFL+`, `L` → `+RF-LFL-FR+`, and `R` → `-LF+RFR+FL-`. Note that Moor's version of the Hilbert's space-filling curve starts and ends at the same point $(\frac{1}{2}, 0)$.

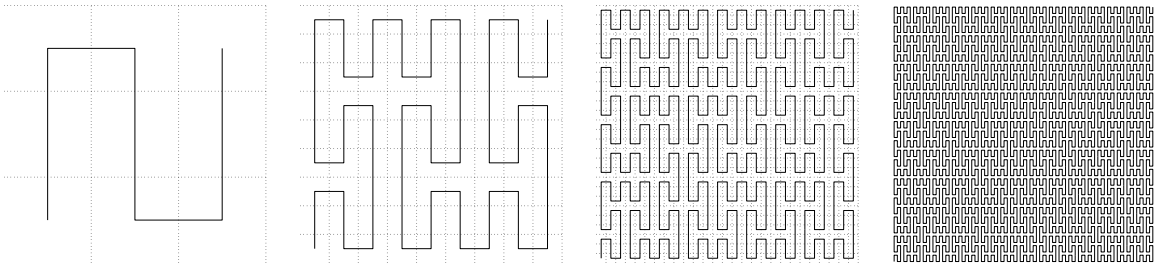
The Hilbert curve can be generalized to n dimensions.[7] A three-dimensional Hilbert curve is shown in Figure 3-8 (c). An L-system with an axiom `X` and a rule `X` → `∧<XF∧<XFX-F∧>>XFX∨F+>>XFX-F>X->` is used for this curve [69].

The Hilbert curve can be also constructed using a *Gray code*, where a sequence

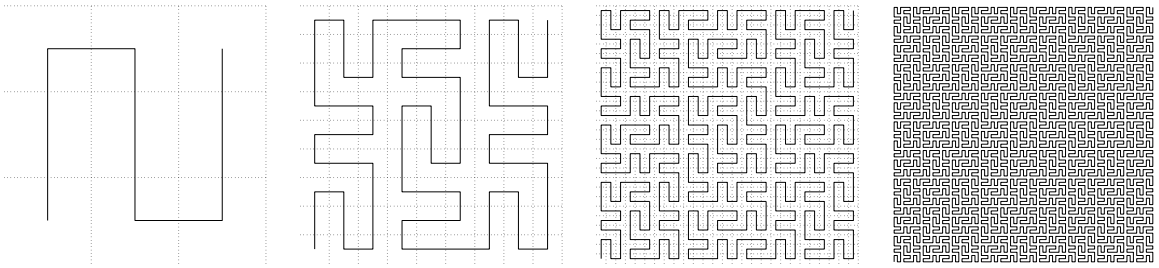
[5]These curves are not fractals in a sense that their fractal dimension n is the same as their topological dimension n, but many space-filling curves are self-similar.

[6]The Peano curve (a) is also of the switch-back type and there are 272 distinct Peano curves of the switch-back type. However, there is only one distinct Peano curve of the meander type [58].

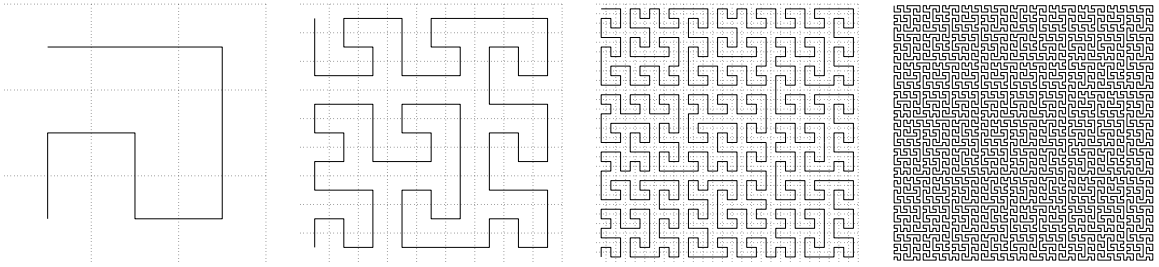
[7]The Peano curve can be generalized to n dimensions, too [58].



(a)

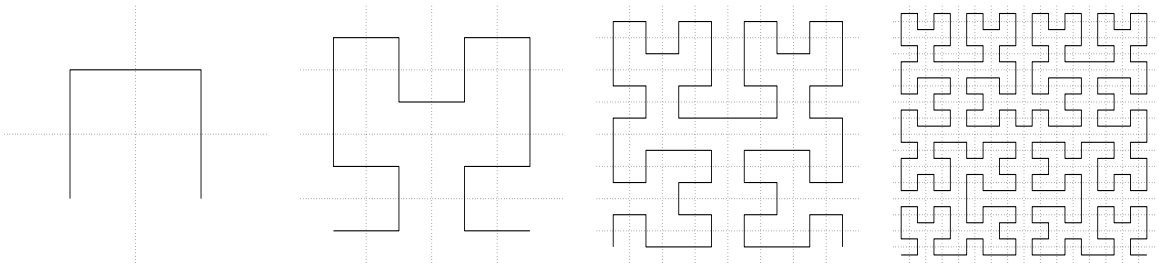


(b)

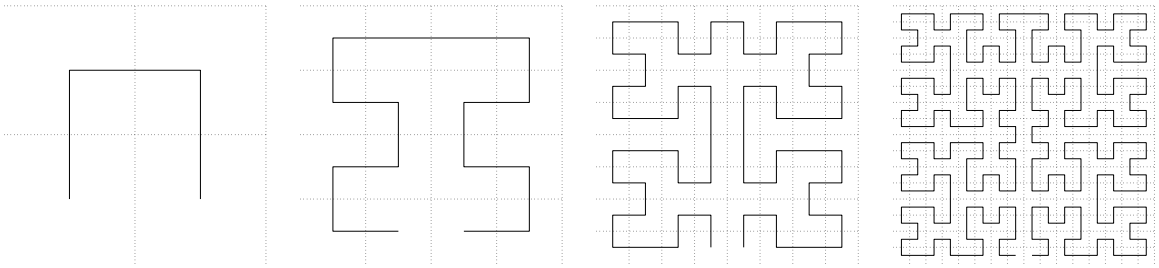


(c)

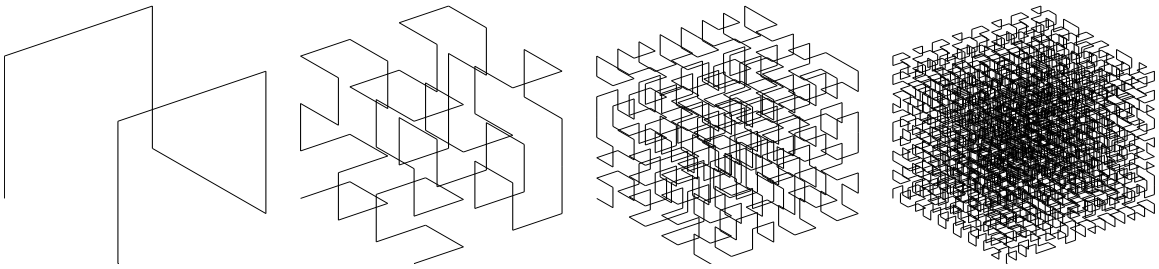
Figure 3-7: Generation of (a) original, (b) switch-back type, and (c) meander type Peano curves



(a)



(b)



(c)

Figure 3-8: Generation of the (a) 2-dimensional, (b) Moor's version, and (b) 3-dimensional Hilbert curves

of length n binary vectors is arranged so that only one bit changes from one vector to the next. In the Hilbert curve, this leads to the *locality-preserving* property of the curve, i.e., as we move along the curve by a small amount, our position on I^2 will not be changed much. In the limit, where the Hilbert curve is defined, this makes the limit curve continuous.

As another example, we define a *hexagonal space-filling curve* by an L-system with an axiom `X` and a set of rules `X` $\rightarrow$ `XF+Y+F+YF-X-F-XFX-FY+` and `Y` $\rightarrow$ `-XF+YFY+F+Y+FX-F-X-FY`. This curve is a variation of the *Peano-Gosper curve*. The difference is that the hexagonal space-filling curve has the nice property that each node is at the center of a hexagon as shown in Figure 3-9 and these hexagons together form a larger hexagon-like area. This makes the curve ideal for quantization, since we can think of these nodes as centroids of a lattice vector quantizer. As in the Peano or Hilbert curves, the hexagonal space-filling curve also has the locality-preserving property.

The boundary of this fractal has a fractal dimension $\log_{\sqrt{7}} 3 \sim 1.129$, since $N = 3$ and $s = \sqrt{7}$. The hexagon-like region $\mathcal{H}$ with the fractal boundary covered by the hexagonal curve is a two-dimensional subset of the plane and is the closure of a fundamental region $\mathcal{R}(A_2)$ of the hexagonal lattice A_2.[8] Therefore, the fractal dimension of $\mathcal{H}$ covered by the hexagonal space-filling curve is two. Although, this is not a space-filling curve as defined previously, this is in fact space-filling since it fills the interior of $\mathcal{H}$.

$\mathcal{H}$ can be subdivided into seven pieces which are all scaled-down images of $\mathcal{H}$. If we consider $\mathcal{H}$ as the closure of a fundamental region $\mathcal{R}(A_2)$ of the hexagonal lattice A_2, then each of the seven subsets of $\mathcal{H}$ is the closure of a fundamental region $\mathcal{R}(\frac{1}{\sqrt{7}}A_2)$ of the finer lattice $\frac{1}{\sqrt{7}}A_2$. In general, we cannot have this property for Voronoi regions; i.e., we cannot take a union of the Voronoi regions of a lattice Λ to construct the Voronoi region of a sublattice Λ' of Λ.

There are some applications where the locality-preserving property of space-filling curve is desirable. One of the most popular uses of space-filling curves is in scanning multi-dimensional data for the purpose of image processing [67, 51, 31, 40] or image

[8] After scaling and rotation.

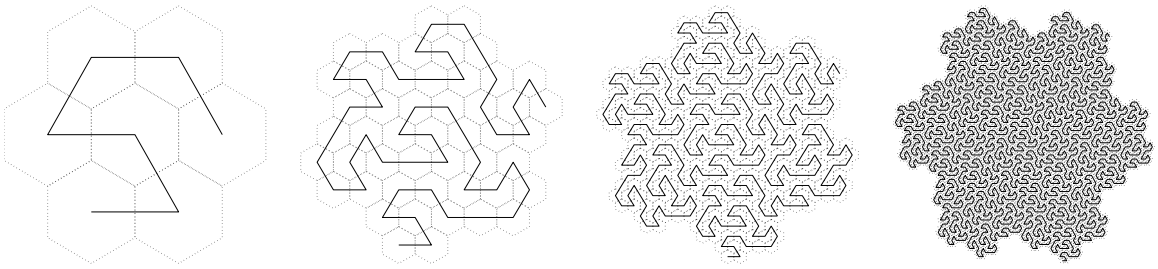


Figure 3-9: Generation of the hexagonal space-filling curve

data compression [43]. A less known application is bandwidth reduction [6], where a multi-dimensional analog source is mapped into a one-dimensional interval using a space-filling curve.

Since space-filling curves are continuous, they are good candidates for compressing a uniform n-dimensional source into a one-dimensional interval, sending the compressed signal through a noisy channel, and recovering the original signal. Although, all space-filling curves constructed using an L-system have the property that as we move one unit on the curve, we move one unit on I^n, it is not easy to see how much average distortion in the source will be caused by the channel noise.

Surprisingly, there has been little understanding in this direction. To date, several attempts have been made to define measures of locality and in some cases to design space-filling curves in [48, 67, 51, 32]. Some authors [67, 32] conclude that the Hilbert curve is good compared to some other curves under their locality measures. However, their locality measures such as the average ratio of the distance between two points in I over the corresponding distance in I^2 are not general enough to handle our distortion measure and the channel noise.

In the next section, we will use space-filling curves for compressing an n-dimensional uniform source defined on a fundamental region of an n-dimensional lattice to the one-dimensional interval I. We assume the compressed signal is corrupted by channel noise, where the channel is mod-$\mathbb{Z}$. Our goal is to construct a system so that the overall distortion is minimized. However, since it is very difficult to find the expected distortion analytically, we will use numerical simulations to find the expected distor-

tion with an arbitrary accuracy. Our results will reveal some interesting aspects of space-filling curves.

3.3 Joint Coding for Uniform Sources

In this section, we construct joint source-channel coding schemes for a uniform source $\mathbf{W}$ on a fundamental region $\mathcal{R}(\Lambda_s)$ of the source lattice Λ_s with a modulo-squared-distortion measure defined on the Voronoi region $\mathcal{R}_V(\Lambda_s)$ of Λ_s and a mod-$\mathbb{Z}$ channel with a truncated Gaussian noise Z defined on the Voronoi region $\mathcal{R}_V(\mathbb{Z})$ of $\mathbb{Z}$. From now on, we use J to denote $\mathcal{R}_V(\mathbb{Z}) = [-\frac{1}{2}, \frac{1}{2}]$. We assume the channel output space is I.[9]

We will consider $\mathbb{Z}^2$, $\mathbb{Z}^3$, or A_2 for the source lattice Λ_s. We consider J^2, J^3, or the hexagon-like fundamental region $\mathcal{H}$ in Figure 3-9 as the source space $\mathcal{W}$.[10] Note the shape of $\mathcal{H}$ changes as we increase the number of points in Figure 3-9. In this case, we may use the Voronoi region $\mathcal{R}_V(A_2)$ of A_2 as the source space by using a modulo mapping, since this does not change the modulo distortion.

We will consider two types of coding schemes in this section. These schemes are based on finite approximations, or approximate curves, for space-filling curves. These approximate curves appeared in Figures 3-7, 3-8, and 3-9 as intermediate steps in generating space-filling curves.

In the first scheme, we form a lattice vector quantizer by using the nodes of an approximate curve as centroids. For example, in Figures 3-7, the approximate curves in the first column have nine nodes. Each of these nine nodes is a centroid of a small square represented by dashed lines. We then send the discrete index on the curve through the channel.

In the second scheme, we use an approximate curve to quantize the source, i.e., we find a point on the curve that is closest to the source point and send the continuous index through the channel.

[9] More rigorously, this is $[0, 1)$. We will use I if there is no confusion.

[10] These source spaces are not fundamental regions. However, they do not pose any problem since the source is uniform.

We will describe these schemes in detail and show some results in the following two sections.

3.3.1 Joint Coding Using Lattice Vector Quantizers

Let L denote the number of nodes in an approximate curve for a space-filling curve, where each node is indexed from 0 to $L-1$ sequentially.[11]

The encoding is done by quantizing the two or three-dimensional source $\mathbf{W}$ by using the L nodes as quantization points, i.e., the source is approximated to the closest node point on the curve. In this case, quantization cells are represented by dashed lines in Figures 3-7, 3-8, and 3-9.

Figure 3-10 (a) shows how encoding is done using a 64-node approximate curve for the Hilbert curve ($L = 64$). The source vector a lies in the quantization cell of which b is the centroid. We assume the channel input x is given by the following:

$$x = \frac{i}{L},$$

where $0 \leq i \leq L-1$ is the index of the node b. For example, if $(-7/16, -7/16)$ is the starting point 0 and if the index increases in the direction toward b, then $i = 10$ in Figure 3-10 (a).

For a uniform source on $\mathcal{R}(A_2)$, we will use the average distortion for the two-dimensional source. For the other cases, we will consider both the component distortion for each source component and the average distortion for the n-dimensional source.

We define the signal-to-distortion ratio (SDR) as the ratio of the source power S over the distortion D, where S is $\frac{1}{12}$ for a uniform source on I and is $\frac{5}{36}$ for a uniform source on $\mathcal{R}_V(A_2)$.[12] The range of the distortion D is always $[0, S]$ for each type of the source. Therefore SDR is always greater than or equal to one.

[11] There is no difference in performance in different indexing (different starting point or different direction) because of the uniformity of the source and the symmetry of the channel.

[12] The distance between two neighbor points is one. Therefore, $V(A_2) = \frac{\sqrt{3}}{2}$.

In Figures 3-7 and 3-8, the horizontal axis is W_1 and the vertical axis is W_2. For the three-dimensional Hilbert curve, W_1, W_2, and W_3 correspond to the first, second, and third columns of the following $L = 8$ example, respectively, whose rows are labeled from 0 to $L-1$ from top to bottom, respectively. Approximate curves for higher L can be generated recursively as shown in Figure 3-8 (c).[13]

$$\begin{pmatrix} -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$$

The channel noise Z is truncated Gaussian and has the following density:

$$p_N(z) = \begin{cases} c\, e^{-\mu z^2}, & \text{if } z \in [-\frac{1}{2}, \frac{1}{2}]; \\ 0, & \text{otherwise,} \end{cases}$$

where $N \in [0, \frac{1}{12}]$ is the noise power and $c, \mu \in \mathbb{R}$ satisfy the normalization and

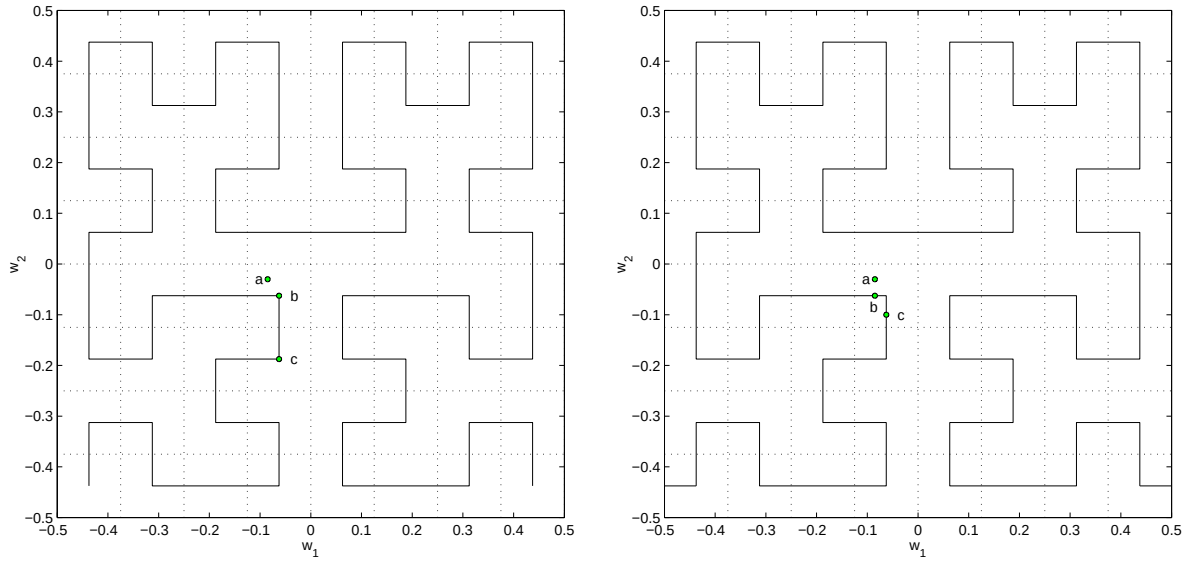
$$\int_{-1/2}^{1/2} z^2 p_N(z)\, dz = N.$$

The channel SNR is defined as the signal power ($P = \frac{1}{12}$) over N. Therefore, the SNR is always greater than or equal to one.

In this section, we quantize the channel output Y into L levels in a symmetric way, e.g., $y \in [-\frac{1}{2L}, \frac{1}{2L})$ is quantized to 0. A more general case, where the channel output is not quantized, is treated in the next section.

In this section, we consider two types of decoders. The first type is the inverse-

[13]The three-dimensional Hilbert curve in Figure 3-8 is plotted using coordinates $(x, y, z) = (-w_2, w_1, -w_3)$ for easy viewing.



(a) Using a lattice vector quantizer

(b) Using an approximate curve with an extended line segment

Figure 3-10: Encoding for uniform sources. (a) source vector, (b) transmitted offset, (c) received offset.

mapping decoder, which is a mapping from the the quantized channel output Y to $\mathcal{W}$ by using the approximate curve that was used to quantize the source. For example, in Figure 3-10 (a), c is the quantized channel output on the curve and is also the reconstructed point.

The second type is a minimum modulo-mean-square error estimator. We will simply call this a modulo-MMSE estimator, or simply an MMSE estimator if there is no confusion. Unlike the usual MMSE estimator, this modulo-MMSE estimator cannot be written as an explicit form. However, at mid to high SNRs, we can find a very good approximation to this, as we show here.

Our goal is to minimize the distortion given the channel output y, i.e.,

$$\min_{\hat{\mathbf{W}}} E[(\hat{\mathbf{W}} - \mathbf{W} \bmod_V \Lambda_s)^2 | Y = y], \tag{3.1}$$

where $(\hat{\mathbf{W}} - \mathbf{W}) \bmod_V \Lambda_s \in \mathcal{R}_V(\Lambda_s)$. As the noise power tends to zero, the error vector $(\hat{\mathbf{W}} - \mathbf{W}) \bmod_V \Lambda_s$ will be concentrated in a small open set $\mathcal{B}$ containing $\mathbf{0}$

with a probability approaching one. Therefore, we can find an n-dimensional vector $\mathbf{h}$ as a function of the channel output y such that both $\Pr((\hat{\mathbf{W}} - \mathbf{h}) \bmod_V \Lambda_s \in \mathcal{B})$ and $\Pr((\mathbf{W} - \mathbf{h}) \bmod_V \Lambda_s \in \mathcal{B})$ approach one as the noise power tends to zero.

Therefore, by differentiating the expectation in (3.1) with respect to $\hat{\mathbf{W}}$, we get a necessary condition for optimality, i.e.,

$$\hat{\mathbf{W}}(y) = (E[(\mathbf{W} - \mathbf{h}) \bmod_V \Lambda_s | Y = y] + \mathbf{h}) \bmod_V \Lambda_s \tag{3.2}$$

with a probability approaching one as the noise power tends to zero. In practice, we can use the output of the inverse-mapping decoder for $\mathbf{h}$, which seems to be a natural choice and actually works very well.

This can be further simplified by realizing that each node point of the approximate curve is the centroid of a quantization cell and that we only need to sum the contributions from those node points when we calculate the conditional expectation $E[(\mathbf{W} - \mathbf{h}) \bmod_V \Lambda_s | Y = y]$, i.e.,

$$\begin{aligned}
&E[(\mathbf{W} - \mathbf{h}) \bmod_V \Lambda_s | Y = y] \\
&\quad = \int_{\mathcal{W}} ((\mathbf{w} - \mathbf{h}) \bmod_V \Lambda_s)\, p(\mathbf{w}|y)\, d\mathbf{w} \\
&\quad = \int_{\mathcal{W}} ((\mathbf{w} - \mathbf{h}) \bmod_V \Lambda_s)\, \alpha\, p(y|\mathbf{w})\, d\mathbf{w} \\
&\quad = \int_{\mathcal{W}} ((\mathbf{w} - \mathbf{h}) \bmod_V \Lambda_s)\, \alpha\, p(y|x(\mathbf{w}))\, d\mathbf{w} \\
&\quad \sim \sum_{i \in K(\mathbf{h})} \int_{\mathcal{W}_i} ((\mathbf{w} - \mathbf{h}) \bmod_V \Lambda_s)\, \alpha\, p(y|x = (i/L))\, d\mathbf{w},
\end{aligned}$$

where $\alpha = p(\mathbf{w})/p(y)$ is a constant since $\mathbf{W}$ is uniform, $x(\mathbf{w})$ is the encoder output for $\mathbf{w}$, $\mathcal{W}_i$ is the i-th quantization cell,[14] and i is the index on the approximate curve. We assume $K(\mathbf{h})$ is a large enough set of indices that includes the index $i(\mathbf{h})$ of $\mathbf{h}$,[15]

[14] We assume these quantization cells are fundamental regions of a finer lattice with ties broken arbitrarily. Therefore, $\mathcal{W}$ is the union of these quantization cells.

[15] We assume $\mathbf{h}$ is the output of the inverse-mapping decoder, i.e., $\mathbf{h}$ is a node point.

i.e.,

$$K(\mathbf{h}) = \{\, (i(\mathbf{h}) + j) \mod L \mid -M \le j \le M\},$$

where M is large enough so that $p(y|x = (i/L))$ becomes negligible if $i \notin K(\mathbf{h})$.

Note that all $\mathcal{W}_i$'s are connected in $\mathbb{R}^n$, where n is the dimension of Λ_s, for all our approximate curves and source spaces J^2, J^3, or $\mathcal{H}$. The integral $\int_{\mathcal{W}_i}(\mathbf{w} - \mathbf{h}) \bmod_V \Lambda_s \, d\mathbf{w}$ is equal to $(\int_{\mathcal{W}_i} \mathbf{w} \, d\mathbf{w} - \frac{V(\mathcal{W})}{L}\mathbf{h}) \mod_V \Lambda_s$ provided that the modulo mapping is equivalent to adding a constant vector for all $\mathbf{w}$'s in $\mathcal{W}$. We assume this holds for all $i \in K(\mathbf{h})$, since we are assuming that $\Pr((\mathbf{W} - \mathbf{h}) \bmod_V \Lambda_s \in \mathcal{B})$ approaches one as the noise power tends to zero.

Unfortunately, this assumption does not hold for approximate curves for Hilbert's space-filling curves with $L = 4$ and for the approximate curve for the three-dimensional Hilbert curve with $L = 8$, unless we assume $\mathbf{W}$ always lies in the same cell as $\mathbf{h}$, in which case the modulo MMSE becomes equivalent to the inverse-mapping decoder.

By defining $\mathbf{h}_i = \frac{\int_{\mathcal{W}_i} \mathbf{w} \, d\mathbf{w}}{V(\mathcal{W})/L}$, where $V(\mathcal{W})$ is the volume of $\mathcal{W}$ and $V(\mathcal{W})/L$ is the volume of $\mathcal{W}_j$ for all $0 \le j \le L - 1$, we get

$$\begin{aligned} &E[(\mathbf{W} - \mathbf{h}) \bmod_V \Lambda_s | Y = y] \\ &\quad \sim \sum_{i \in K(\mathbf{h})} ((\mathbf{h}_i - \mathbf{h}) \bmod_V \Lambda_s) \, \alpha \, p(y|x = (i/L)) \, \frac{V(\mathcal{W})}{L} \\ &\quad = \sum_{i \in K(\mathbf{h})} ((\mathbf{h}_i - \mathbf{h}) \bmod_V \Lambda_s) \, \alpha \, c e^{-\mu((y - i/L) \bmod_V \mathbb{Z})^2} \, \frac{V(\mathcal{W})}{L}, \end{aligned}$$

where $p(y|x) = ce^{-\mu((y-x) \bmod_V \mathbb{Z})^2}$ and $((y - x) \bmod_V \mathbb{Z}) \in J$.

Appendix B shows the performance results of some space-filling curves in Figures 3-7, 3-8, and 3-9. Table 3.2 summarizes these results.

There is no big difference in performance of each curve. Two-dimensional Hilbert curves perform the best among the curves used in Table 3.2. When the MMSE decoder is used, we can see some slight improvement in performance, but not much. This means that these approximate curves do not cause too much bias.

Table 3.2: The SDR gap in dB from the Shannon limit for uniform sources and 2:1, 3:1 B/W reduction using some space-filling curves. Mid-SNR performance is measured at SNR = 50 dB. Best performance (closest to the Shannon limit) is measured for high SNR > 20 dB.

		Inverse-mapping			MMSE		
Curve	L	at 50 dB	best	at [dB]	at 50 dB	best	at [dB]
Peano	6561	3.9	2.5	77	2.6	2.1	77
switch-back type	6561	3.5	2.4	78	2.3	2.1	76
meander type	6561	3.5	2.4	78	2.2	2.1	76
Hilbert	4096	3.3	2.4	74	2.0	1.9	51
Moor's version	4096	3.3	2.4	74	2.0	1.9	45
3D Hilbert	4096	3.7	2.2	74	2.1	2.0	73
Hexagonal	2401	3.9	2.4	69	2.5	2.2	68

Figures B-1–B-6 show plots of SDRs as functions of the channel SNR for various approximate curves. The performance of curves with small L also shows similar behavior as the fine quantization results in Table 3.2 except that they show flatness at lower SDRs.

Except for the two-dimensional Hilbert curves, SDRs for each source component are quite different, suggesting that each space-filling curve has quite different fine structures. For example, when approximate curves for the Peano curve are used, the SDR for W_1 is larger than that of W_2 at mid-SNR regions. This is due to the fact that vertical line segments are usually longer than horizontal segments in approximate curves for the Peano curve.

Since the quantization part of this system is a lattice quantizer, the saturated SDR ($\mathrm{SDR}_{\mathrm{sat}}$) can be directly determined from L, i.e.,

$$\mathrm{SDR}_{\mathrm{sat}} = L^{\frac{1}{n}},$$

where n is the dimension of the source. MMSE estimation does not improve the saturated SDR, due to the symmetry in the quantization and the uniformity of the source.

Figures B-1–B-6 suggest that the asymptotic SDR gap from the Shannon limit

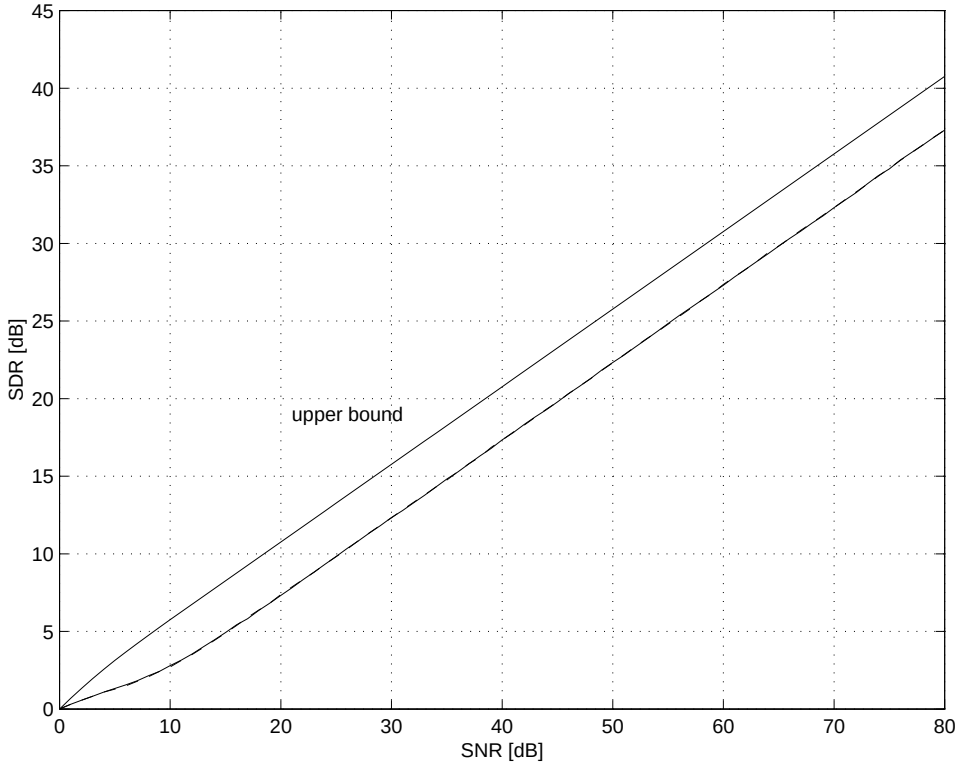


Figure 3-11: 2:1 bandwidth reduction using the Hilbert curve approximated using a 52-bit number for uniform sources. SDRs for W_1 (—) and W_2 (- -) are plotted as functions of the channel SNR.

might be constant as L tends to infinity. In Figure 3-11, we show simulation results using the full precision of the 64-bit double precision number,[16] i.e., $L \sim 2^{52}$, for approximating the Hilbert curve. The asymptotic SDR gap is about 3.5 dB and there is no noticeable difference between the SDRs of the two source components. The asymptotic SDR gap should be constant due to the self-similar nature of space-filling curves.

Figure 3-12 shows the Hilbert curve mapped on a torus to signify that the distortion measure is modulo.

[16]Fraction part is 52-bit long.

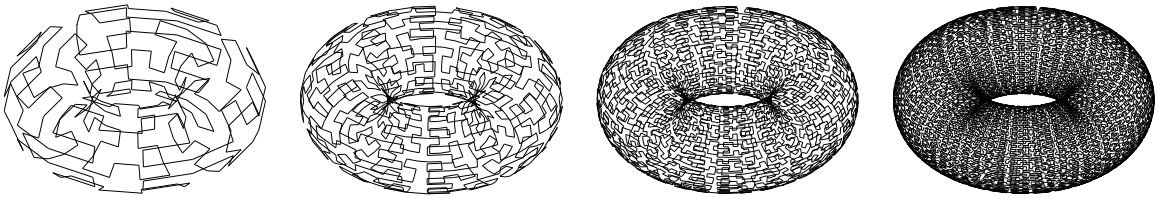


Figure 3-12: Generation of the Hilbert curve mapped on a torus

3.3.2 Joint Coding Using Approximate Curves

In this section, we consider a coding scheme similar to that presented in the previous section. It is different in that we use a continuous approximate curve for a space-filling curve instead of a set of node points of the approximate curve. We will use the same notations here as in the previous section.

Since the channel input is continuous (and modulo) in this case, we need to make an approximate curve closed for symmetry. To do this, we attach a line segment to each end point of an approximate curve to the boundary of the source space such that these two line segments meet at one point up to mod-Λ_s. We call this segment pair an extended line segment. We choose the extended line segment such that its length is minimized. Note that for Moor's version of Hilbert's space-filling curve, the extended line segment is simply a line between two end points.

For the hexagonal space-filling curve, there is ambiguity since there are two ways of choosing this extended line segment for its approximate curves. However, at high SNR, this should not make much difference since it does not contribute much to the distortion.

For our space-filling curve examples (except for Peano curves of the switch-back type), the length of this extended line segment is the same as the length of any line segments between two neighboring nodes in an approximate curve. For approximate curves for Peano curves of the switch-back type, however, the extended line segment is $\sqrt{2}$ times longer than the other line segments, since two end points in the approximate curve are on the diagonal of J^2. However, this should not be a problem at high SDRs,

since the contribution from those two cells becomes small.

For encoding, we first find a quantization cell that contains the source vector. Then, we determine the point t on the curve that is closest to the source vector, and we calculate the channel input x by the following:

$$x = \frac{i + \alpha}{L},$$

where i and $(i+1) \bmod L$ are the two adjacent nodes connected by the line segment containing t and $\alpha \in [0, 1)$ is the relative offset of t from the node i toward the next node $(i+1) \bmod L$. For example, in Figure 3-10 (b), b is the point on the curve that is the closest to the source vector a.

Note that this is not the best thing we can do since the encoded point t may not be the closest point on the curve to the source vector, i.e., the closest point may be in a neighbor quantization cell. However, this seems to make analysis too complicated without much improvement in performance, especially for space-filling curves on I^2.[17] Furthermore, this complicates the modulo-MMSE decoding, which otherwise has a simple formulation as we will discuss later. Therefore, we will consider only the case in which the encoder's search is limited to the quantization cell of the source vector.

We also consider two types of decoders in this section. The first type is the inverse-mapping decoder where the channel output is mapped to the decoded source vector using an approximate curve. For example, in Figure 3-10 (b), c is the channel output mapped on the curve and is also the reconstruction point.

The second type is a modulo-MMSE decoder. It works the same way as the modulo-MMSE decoder worked in the previous section, except now we need to compute n-dimensional integrals to calculate the conditional expectation $E[(\mathbf{W}-\mathbf{a}) \bmod_V$

[17] For the hexagonal space-filling curve, we have observed minor improvement when we allow the encoder to look at line segments in neighbor cells also.

$\Lambda_s|Y = y]$, which can be calculated similarly as in the previous section, i.e.,

$$\begin{aligned}
&E[(\mathbf{W}-\mathbf{h}) \bmod_V \Lambda_s|Y=y] \\
&\quad = \int_{\mathcal{W}} ((\mathbf{w}-\mathbf{h}) \bmod_V \Lambda_s)\, p(\mathbf{w}|y)\, d\mathbf{w} \\
&\quad = \int_{\mathcal{W}} ((\mathbf{w}-\mathbf{h}) \bmod_V \Lambda_s)\, \alpha\, p(y|\mathbf{w})\, d\mathbf{w} \\
&\quad = \int_{\mathcal{W}} ((\mathbf{w}-\mathbf{h}) \bmod_V \Lambda_s)\, \alpha\, p(y|x(\mathbf{w}))\, d\mathbf{w} \\
&\quad \sim \sum_{i\in K(\mathbf{h})} \int_{\mathcal{W}_i} ((\mathbf{w}-\mathbf{h}) \bmod_V \Lambda_s)\, \alpha\, p(y|x(\mathbf{w}))\, d\mathbf{w} \\
&\quad = \sum_{i\in K(\mathbf{h})} \int_{\mathcal{W}_i} ((\mathbf{w}-\mathbf{h}) \bmod_V \Lambda_s)\, \alpha\, ce^{-\mu((y-x(\mathbf{w})) \bmod_V \mathbb{Z})^2}\, d\mathbf{w},
\end{aligned}$$

where we use the same $K(\mathbf{h})$, α, and $\mathcal{W}_i$ as defined in the previous section.

Assuming the modulo mapping in $(\mathbf{w}-\mathbf{h}) \bmod_V \Lambda_s$ is equivalent to adding a constant vector as done in the previous section, we get

$$\begin{aligned}
&E[(\mathbf{W}-\mathbf{h}) \bmod_V \Lambda_s|Y=y] \\
&\quad \sim \sum_{i\in K(\mathbf{h})} \Bigg[\int_{\mathcal{W}_i} \mathbf{w}\, \alpha\, ce^{-\mu((y-x(\mathbf{w})) \bmod_V \mathbb{Z})^2}\, d\mathbf{w} \\
&\qquad\qquad - \mathbf{h} \int_{\mathcal{W}_i} \alpha\, ce^{-\mu((y-x(\mathbf{w})) \bmod_V \mathbb{Z})^2}\, d\mathbf{w} \Bigg] \bmod_V \Lambda_s.
\end{aligned}$$

For $\mathcal{W} = J^2$, this can be done efficiently since there are only two possibilities[18] for a curve to pass a node point, namely, a straight-through and a 90°-turn as shown in Figure 3-13.[19] For these two patterns, we can construct a table lookup for each noise power N and for each error $y - x(\mathbf{w})$, where we assume fine quantizations for both the noise power and the error. Sometimes, these two patterns need to be rotated. Since this can be done by changing the coordinate (w_1, w_2) appropriately, we need only two table lookups. Finally, we can calculate $\hat{\mathbf{W}}(h)$ by summing the above integrals and

[18]For Peano curves of the switch-back type, two cells that contain the end points have 135°-turns. However, these can be approximately done assuming they are straight-throughs. In practice, this should not cause a noticeable degradation.

[19]It is slightly more complicated, but can be done similarly for $\mathcal{W} = J^3$ and $\mathcal{H}$.

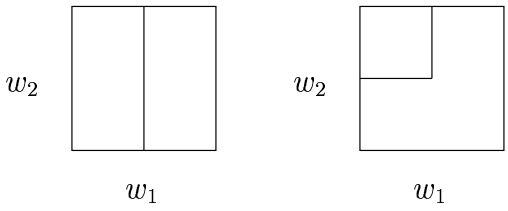


Figure 3-13: Two possible patterns for modulo-MMSE decoding for uniform sources on J^2

using (3.2).

In Appendix C, we show the simulation results for the coding schemes in this section. We summarize the results in Table 3.3 and 3.4.

Improvements in mid SNRs are minor compared to the coding scheme in the previous section. However, the peak performance for both the inverse-mapping and MMSE decoders is greatly improved compared to the coding schemes based on the lattice vector quantizers. Note that MMSE decoders improve the mid-SNR performance by more than 1 dB, but by only about $0.1 \sim 0.2$ dB at the peak.

Saturated SDR, i.e., the SDR when the channel noise is zero, for the coding scheme in the previous section was $L^{1/n}$. We observe that the saturated SDR is greatly improved by using an approximate curve combined when the inverse-mapping decoding is used. For example, when the channel noise zero, the quantization cell with a straight line in Figure 3-13 will reduce the average distortion by half (about 3 dB) compared to the quantization to a single point in the center of the square. The 90°-turn cell in Figure 3-13 will reduce the average distortion by $\frac{5}{8}$ (about 2 dB), since it becomes the lattice quantizer with probability $\frac{1}{4}$ and becomes the straight line quantizer in Figure 3-13 with probability $\frac{3}{4}$. Therefore, the actual saturated SDR will be improved by about $2 \sim 3$ dB compared to the scheme based on a lattice quantizer, which is in excellent agreement with simulation results. When modulo-MMSE decoding is used, even more improvement in the saturated SDR is observed.

We also observe that the saturated SDRs for W_1 and W_2 are very different except in the Hilbert curve case. Interestingly, the two performances are reversed at low-SNR regimes. This is because long vertical or horizontal segments are bad for the corresponding source component at low SNR because the source can be corrupted

Table 3.3: The SDR gap in dB from the Shannon limit for uniform sources and 2:1 B/W reduction using approximate curves for several space-filling curves. Mid-SNR performance is measured at SNR = 50 dB. Best performance (closest to the Shannon limit) is measured for high SNR > 20 dB.

		Inverse-mapping			MMSE		
Curve	L	at 50 dB	best	at [dB]	at 50 dB	best	at [dB]
Peano	6561	3.8	1.2	76	2.6	1.1	77
switch-back type	6561	3.4	1.3	75	2.3	1.1	77
meander type	6561	3.4	1.4	75	2.2	1.2	77
Hilbert	4096	3.1	1.5	71	2.0	1.3	73
Moor's version	4096	3.1	1.4	71	2.0	1.3	74

Table 3.4: The minimum SDR gap in dB from the Shannon limit for uniform sources and 2:1 B/W reduction using several approximate curves for the Peano curve. The gap is measured for high SNR > 15 dB.

	Inverse-mapping		MMSE	
L	SDR	at [dB]	SDR	at [dB]
9	1.4	19	1.3	21
81	1.2	38	1.1	38
729	1.2	56	1.1	57
6561	1.2	76	1.1	77

by the noise more easily, but they are good for quantization at high SNR because it reduces the quantization noise.

The best performing curve is the Peano curve and its performance is only about 1.1 dB from the Shannon limit. This is slightly better than the simulation results in Section 3.1.3.

3.4 Joint Coding for Gaussian Sources

In this section, we design joint coding schemes for Gaussian sources. We first construct joint coding schemes using stretched space-filling curves in the following section and then design coding schemes using spirals and spiral-like curves in Section 3.4.2.

3.4.1 Joint Coding Using Stretched Space-Filling Curves

We extend the idea of bandwidth reduction using space-filling curves of the previous section to an iid Gaussian source and the AWGN channel here. We assume an iid zero-mean Gaussian source $\mathbf{X}$ with variance one.

In this section, we modify the joint coding scheme for uniform sources in Section 3.3 for iid Gaussian sources. The encoder compresses two iid Gaussian inputs (W_1, W_2) using a finite-length two-dimensional curve into one real number by finding the nearest point on the curve and sending the position on the curve $x \in [0, 1]$ through the AWGN channel.

Since space-filling curves used for uniform sources are uniformly distributed, we stretch an approximate curve to fit the Gaussian density.

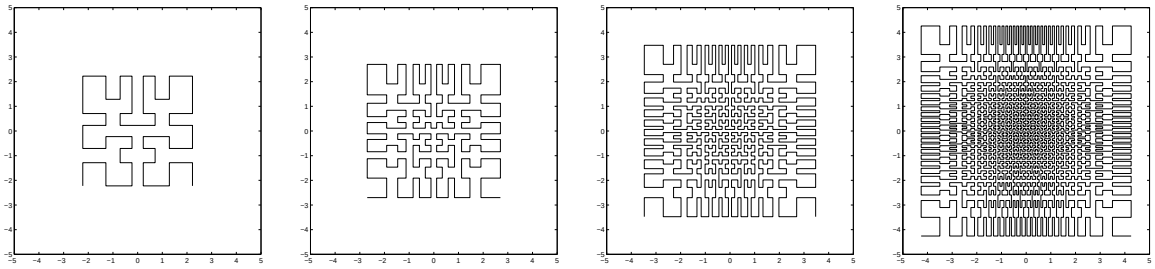
For high-rate quantization, the pdf of the quantization points should be proportional to $p(\mathbf{w})^{1/(1+r)}$ for optimality, where $p(\mathbf{w})$ is the source pdf of $\mathbf{W}$ and r is the exponent in the distortion measure $||\cdot||^r$ [29]. However, since the node points of an approximate curve are not just discrete sampling points but are connected and mapped to the channel input, it is a difficult task to find the optimal stretching function for space-filling curves.

One easy option is to optimize r' where the pdf of the quantization points follow $p(\mathbf{w})^{1/(1+r')}$. Since $p(\mathbf{w})$ is a product of two Gaussian densities, the density of quantization points using $p(\mathbf{w})^{1/(1+r')}$ becomes a product of two Gaussian densities. Therefore, assuming an approximate curve in J^2 is dense enough, we can compute the stretched point (v_1, v_2) of the coordinate (u_1, u_2) of the nodes of the approximate curve using the following mapping:

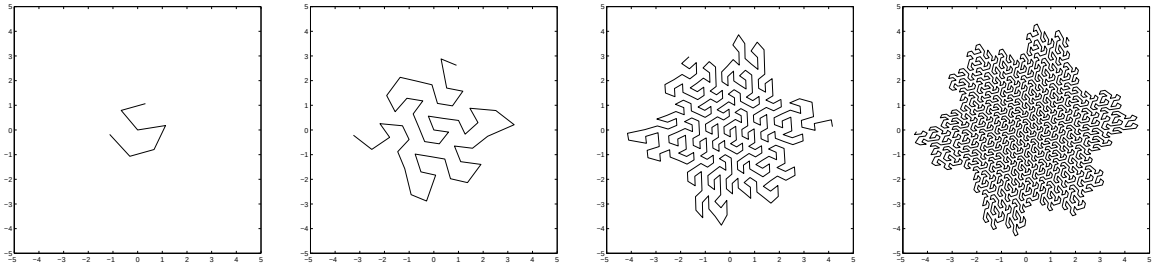
$$v_i = \sqrt{2(1+r')}\,\mathrm{erf}^{-1}(2u_i),\ i = 1, 2.$$

Therefore, (v_1, v_2) follows $p(\mathbf{w})^{1/(1+r')}$ assuming (u_1, u_2) is uniformly distributed on J^2.

To use an approximate curve for the hexagonal space-filling curve, we need to scale the approximate curve first to ensure that the scaled curve is included in J^2. We do



(a) 64, 256, 1024, 4096-point stretched Hilbert curves.



(b) 7, 49, 343, 2401-point stretched hexagonal curves.

Figure 3-14: Stretched curves for Gaussian

this heuristically, i.e., we scale 7-, 49-, 343-, and 2401-point approximate curves by 1.0, 0.9, 0.8, and 0.7, respectively. These numbers are chosen such that node points that are closest to the boundary of J^2 are not too close or too far from the boundary to prevent the curve from being stretched too much or too little, respectively. This is certainly not optimal, but we have observed that the performance does not depend much on this scaling unless it causes too much stretching.

Figure 3-14 shows some examples of stretched curves. r' values are -0.1, 1.1, 1.1, 1.1, 1.6, and 2.1 for 4-, 16-, 64-, 256-, 1024-, and 4096-point approximate curves for the Hilbert curve. We use $r' = 0.0$, 2.4, 5.5, and 10.8 for 7-, 49-, 343-, and 2401-point approximate curves for the hexagonal space-filling curve. r' values are chosen so that the peak performance (the SDR gap from the Shannon limit) at high-SNR is maximized.

Figures 3-15 and 3-16 show simulation results for two iid Gaussian sources and the AWGN channel using the stretched curves of Figure 3-14. Note that as we use more points we get better performance at high SNR, but not at low SNR. The performance curves are about 1 to 7 dB away from the Shannon limit. However, they are almost

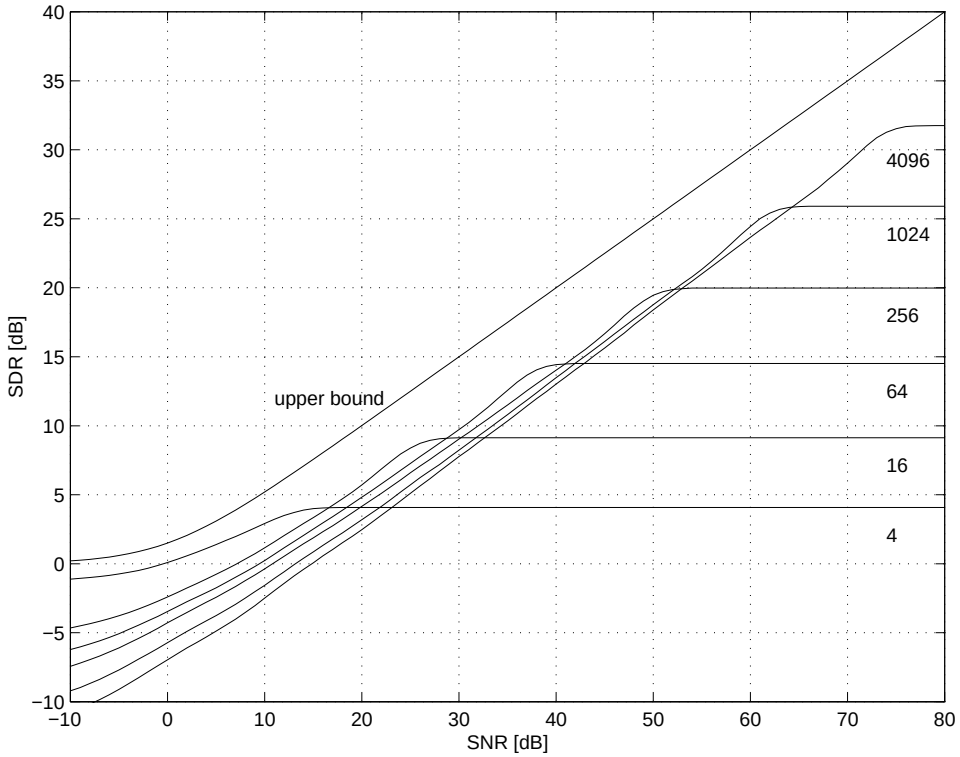


Figure 3-15: 2:1 bandwidth reduction using stretched Hilbert curves for Gaussian (number of nodes of each stretched curve shown)

parallel to the upper bound all the way up to the saturation point, which could be desirable in some applications.

3.4.2 Joint Coding Using Spiral Curves

We now consider different families of curves for two iid Gaussian sources. Without loss of generality, we assume a zero-mean Gaussian distribution with variance σ_w^2. Since such a two-dimensional Gaussian distribution has circular symmetry, it seems plausible to use a curve that exhibits such symmetry. In fact, numerically designed constellations in Section 3.1.3 are very much like spirals as also observed by Fuldseth [24].

We consider spiral curves as shown in Figure 3-17, where a, b, and c represent the source vector, the transmitted offset, and the received offset, respectively. The spiral curve can be described as a parametrized curve, in which $r(x) \in \mathbb{R}^+$ is the radius of

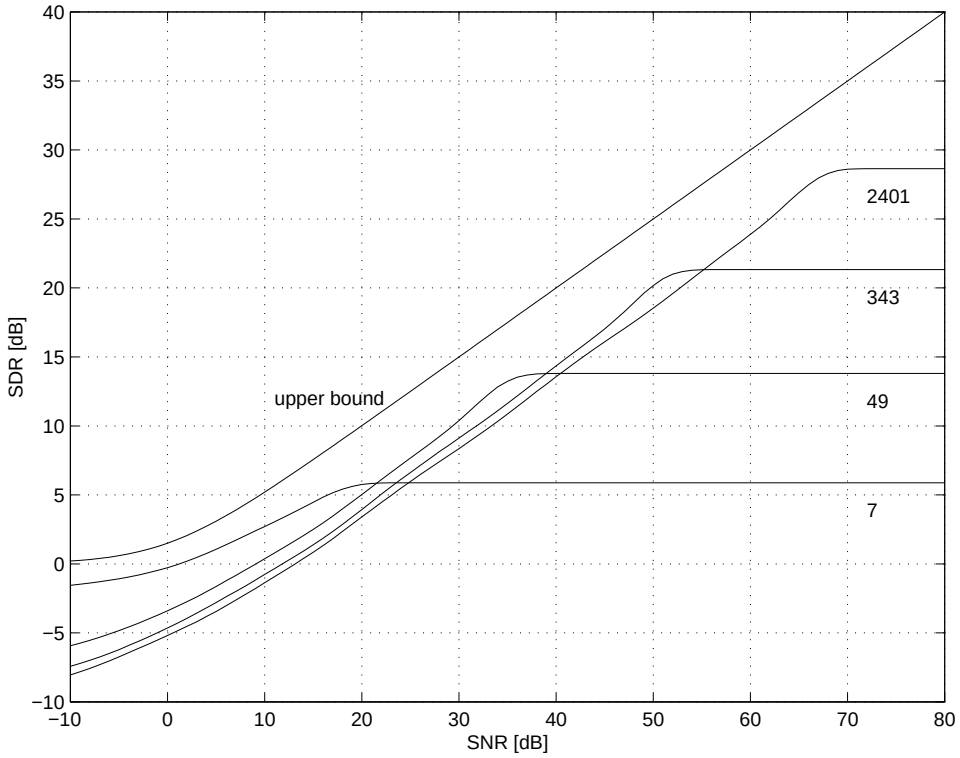


Figure 3-16: 2:1 bandwidth reduction using stretched hexagonal curves for Gaussian (number of nodes of each stretched curve shown)

the curve and $s(x) \in \mathbb{R}^+$ is the radial gap between two spiral arms, where $x \in \mathbb{R}$ is the channel input. The goal is to find the optimal pair $\{r(x), s(x)\}$, where the power constraint is given by $E[x^2] \leq P$ and $r(0) = 0$. This characterization of the curve becomes ambiguous near the origin. However, since we will use this parametrization for high-SNR analysis, the curve becomes dense and the near-origin behavior can be ignored.

A two-dimensional vector $a = (w_1, w_2)$ is encoded as $b = (b_1, b_2)$, where b is the nearest point on the curve to a. Then, we find x and transmit it, where the magnitude of x is $|x| = r^{-1}(|b|)$ and the sign of x depends on what side of the curve b is on.[20] The AWGN z has zero-mean and variance σ_z^2, and is independent of the source (w_1, w_2). The output y of the channel is then mapped to the point $c = (y_1, y_2)$ on the two-dimensional plane via either the inverse mapping or the MMSE estimator.

The expected distortion D per input is then given by $E[||a - c||^2]/2$. Since the optimized spiral will be dense with respect to σ_w at high SNR, we may assume that D can be decomposed into two components $D = D_r + D_\theta$, where $D_r = E[(||a|| - ||c||)^2]/2$ is the expected radial distortion per input and $D_\theta = E[||c||^2(\angle a - \angle c)^2]/2$ is the expected angular distortion per input. D_r comes from quantization, and D_θ comes from the channel noise. As can be seen from the form of decomposition, D_r will be equal to D_θ at the optimum point as we will see later, because such a balance between two distortion measures will minimize the total distortion.

We will consider two families of spiral curves: uniform spiral curves and more general nonuniform spiral curves. First, we define the uniform spiral curve as a spiral with a constant gap $s = s(x)$. We further simplify the problem by assuming that the channel input x is proportional to $r(x)^2$; in other words, as we change the channel input linearly with a constant speed, we correspondingly move at a constant speed on the two-dimensional space following the spiral pattern. The optimization problem

[20] Any side can be positive due to symmetry.

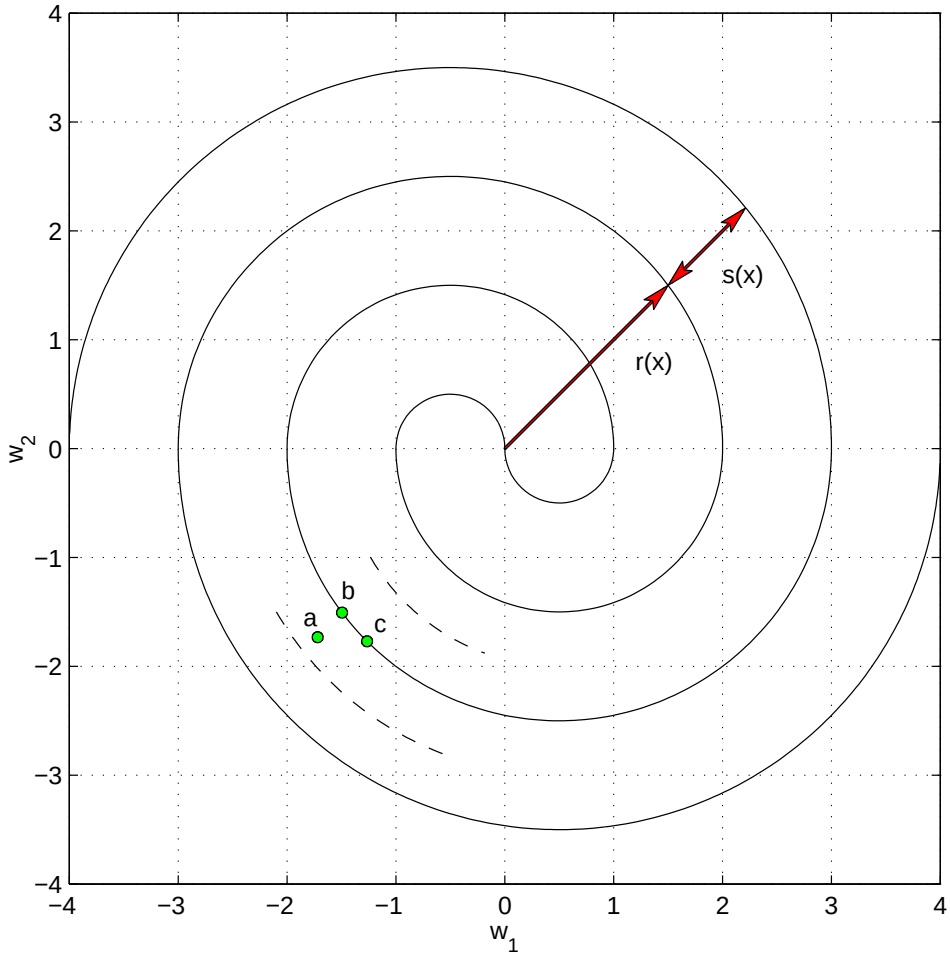


Figure 3-17: Joint coding using a uniform spiral curve for two iid unit-variance Gaussians. (a) source vector, (b) transmitted offset, (c) received offset.

for this case becomes

$$\min_{s:E[x^2]\leq P} D(s),$$

where we minimize the distortion $D = D(s)$ by optimizing the gap s for the given power constraint P. The solution to this problem is given by the following

$$s = \frac{\sigma_w}{2}\sqrt[4]{\frac{3\pi^2}{2\cdot \mathrm{SNR}}},$$

where SNR is given by P/σ_z^2. The pdf $p_X(x)$ of the channel input x has the following form

$$p_X(x) = \frac{\alpha}{2}e^{-\alpha|x|},$$

where α is a function of s.

Since the channel input density is not Gaussian, this system cannot achieve the Shannon limit. However, this simple system is surprisingly good. The SDR defined as σ_w^2/D can be expressed as a function of SNR,

$$\mathrm{SDR} = \frac{\sqrt{6}}{\pi}\sqrt{\mathrm{SNR}}.$$

The Shannon limit for 2:1 compression is given by

$$\mathrm{SDR}_{\mathrm{opt}} = \sqrt{1+\mathrm{SNR}}.$$

Therefore, the asymptotic gap between the SDR the optimal SDR is only about 1.1 dB.

Figure 3-18 shows simulation results for several uniform spiral curves, where each curve is optimized for different channel SNRs. The minimum SDR gap from the rate-distortion limit for the middle two curves is about 1.1 dB as predicted by the above analysis for high SNR. This is slightly better than the simulation results in

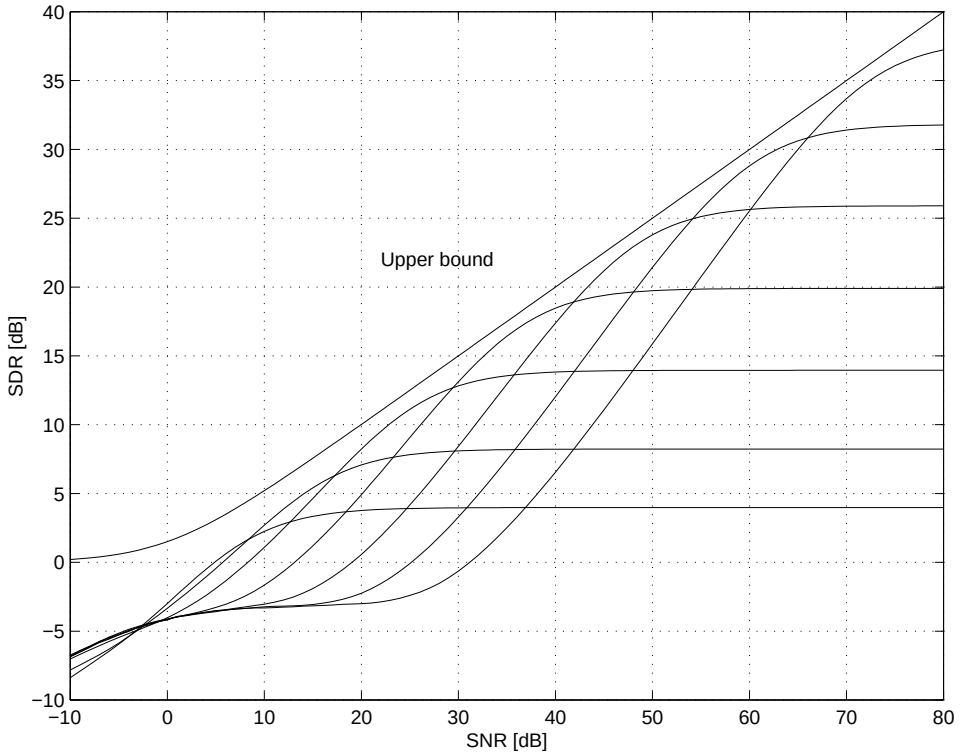


Figure 3-18: 2:1 bandwidth reduction using uniform spiral curves for Gaussian

Section 3.1.3. In the case of a channel mismatch, we observe some degradation. However, the range of SNR required to keep the SDR within 5 dB from the upper bound is very large, about 30 dB, suggesting that this scheme is very robust.

Finally, we generalize the optimization problem by considering any nonnegative functions $r(x)$ and $s(x)$. We assume as before that the distortion D is given by $D_r + D_\theta$. The optimization problem can be expressed as the following:

$$\min_{r(x),s(x):E[x^2]\leq P} D(r(x), s(x)).$$

Solving this problem for $r(x)$ and $s(x)$, we get the following result.

Theorem 3.1 $\{r(x), s(x)\} = \text{argmin}_{r(x),s(x):E[x^2]\leq P} D(r(x), s(x))$ *is given by*

$$\begin{aligned} r(x) &= 2\sigma_w \sqrt{-\log\left[\Gamma\left(\frac{2}{3}\right) 3^{\frac{2}{3}} A_i(\lambda x)\right]}; \\ s(x) &= 2\sigma_w \sqrt{-\sqrt{3}\pi\sigma_z [\log A_i(\lambda x)]'}, \end{aligned}$$

and the pdf $p_X(x)$ of the channel input x is given by

$$p_X(x) = -\Gamma^2\left(\frac{2}{3}\right)3^{\frac{4}{3}}\frac{1}{2}[A_i^2(\lambda x)]',$$

where λ is determined from P, $\Gamma(z)$ is the gamma function, and $A_i(z)$ is the Airy function.

The Airy functions $A_i(z)$ together with $B_i(z)$ are independent solutions of the following differential equation

$$\frac{d^2u}{dz^2} - zu = 0.$$

The solutions to this problem can be expressed using Bessel functions as follows [33]:

$$\begin{aligned} A_i(z) &= \frac{\sqrt{z}}{3}\left\{I_{-\frac{1}{3}}\left(\frac{2}{3}z^{\frac{3}{2}}\right) - I_{\frac{1}{3}}\left(\frac{2}{3}z^{\frac{3}{2}}\right)\right\}, \\ A_i(-z) &= \frac{\sqrt{z}}{3}\left\{J_{\frac{1}{3}}\left(\frac{2}{3}z^{\frac{3}{2}}\right) + J_{-\frac{1}{3}}\left(\frac{2}{3}z^{\frac{3}{2}}\right)\right\} \end{aligned}$$

and

$$\begin{aligned} B_i(z) &= \sqrt{\frac{z}{3}}\left\{I_{-\frac{1}{3}}\left(\frac{2}{3}z^{\frac{3}{2}}\right) + I_{\frac{1}{3}}\left(\frac{2}{3}z^{\frac{3}{2}}\right)\right\}, \\ B_i(-z) &= \sqrt{\frac{z}{3}}\left\{J_{\frac{1}{3}}\left(\frac{2}{3}z^{\frac{3}{2}}\right) - J_{-\frac{1}{3}}\left(\frac{2}{3}z^{\frac{3}{2}}\right)\right\}. \end{aligned}$$

The density of the channel input x can be approximated by

$$p_X(x) = -\Gamma^2\left(\frac{2}{3}\right)3^{\frac{4}{3}}\frac{1}{2}[A_i^2(\lambda x)]' \sim ce^{-\alpha|x|^{1.5}},$$

where α is a function of λ and the exponent 1.5 in the function on the right hand side is obtained (approximately) using a curve fitting. This suggests that this scheme is still not optimal in the Shannon sense since x is not Gaussian. However, it performs

remarkably well. The SDR can be expressed as a function of SNR as follows:

$$\mathrm{SDR} = \frac{\sqrt{3\sqrt{3}\pi}}{2\Gamma^3(\frac{2}{3})}\sqrt{\mathrm{SNR}}.$$

The asymptotic SDR gap is now only about 0.9 dB from the Shannon bound.

3.5 Summary

In this chapter, we have constructed various joint source-channel coding schemes based on space-filling and other families of curves for various sources and channels when the source bandwidth is two or three times that of the channel.

For some uniform sources and modulo channels, we have constructed joint coding schemes that approach the Shannon limit within about 1.1 dB for a wide range of the channel SNR.

For Gaussian sources and the AWGN channel, our joint coding schemes based on spiral-like curves are within about 0.9 dB of the Shannon limit for a wide range of the channel SNR.

Chapter 4

Normalization of Channels

In this chapter, we consider binary-input symmetric-output memoryless channels. We identify a unified approach to these channels, which we call "normalization."

4.1 Channel Model

We define binary-input symmetric-output memoryless channels as:

Definition 4.1 *A* binary-input symmetric-output memoryless channel *(symmetric channel, in short) is a discrete-time channel whose input X is ± 1 and whose output Y (discrete or continuous) depends only on the current input symbol and satisfies the following:*

$$p(Y = y|X = 1) = p(Y = -y|X = -1),$$

where $p(Y|X)$ is the transition probability, $y \in \mathcal{Y}$, and $\mathcal{Y}$ is a subset of $\mathbb{R}^n$, $\mathbb{C}^n$, or a finite group.

Binary-input symmetric-output memoryless channels are "symmetric" in the sense of Gallager [28, 22]. He defined a symmetric channel as a channel with a transition matrix such that the output space can be partitioned into subsets such that for each subset the submatrix of the transition matrix has the property that each column of the submatrix is a permutation of every other column and each row is a permutation

of every other row. Gallager proved that the channel capacity of a symmetric channel can be achieved using equi-probable inputs. Therefore, the capacity-achieving input distribution of a binary-input symmetric-output channel is equi-probable. From now on, we use only binary-input symmetric-output channels. We present many interesting examples of symmetric channels in Section 6.1.

For symmetric channels, channel capacity C is simply given by the following:

Theorem 4.1 *The channel capacity of a binary-input symmetric-output memoryless channel with the transition probability $p(Y|X)$ is given by*

$$C = 1 - E\left[\log_2\left(1 + e^{-u(y)}\right)|X = 1\right], \tag{4.1}$$

where the log-likelihood ratio (LLR) u as a function of the channel output y (which we call a LLR mapping) is defined as

$$u(y) = \log \frac{p(y|X = 1)}{p(y|X = -1)}. \tag{4.2}$$

Proof: The channel capacity of a symmetric channel is given by the following:

$$C = h(Y) - h(Y|X),$$

where X is equiprobable. Since $h(Y|X = 1) = h(Y|X = -1)$ from the symmetry, we get the following:

$$\begin{aligned} C &= h(Y) - h(Y|X = 1) \\ &= -\int \tilde{p}(y) \log_2 \tilde{p}(y)\, dy + \int p(y|X = 1) \log_2 p(y|X = 1)\, dy, \end{aligned}$$

where $h(\cdot)$ is either the entropy or the differential entropy depending on the variable Y and $\tilde{p}(y)$ is given by

$$\tilde{p}(y) = \frac{p(y|X = 1) + p(y|X = -1)}{2}.$$

Using the symmetry again, we get the following:

$$\int \tilde{p}(y) \log_2 \tilde{p}(y)\, dy = \int p(y|X=1) \log_2 \tilde{p}(y)\, dy.$$

Therefore, the channel capacity is given by

$$\begin{aligned} C &= \int p(y|X=1) \log_2 \frac{p(y|X=1)}{\tilde{p}(y)}\, dy \\ &= 1 - \int p(y|X=1) \log_2 \left(1 + \frac{p(y|X=-1)}{p(y|X=1)}\right)\, dy \\ &= 1 - \int p(y|X=1) \log_2 \left(1 + e^{-u(y)}\right)\, dy. \end{aligned}$$

□

From now on, if there is no confusion, we simply write (4.1) as

$$C = 1 - E\left[\log_2 \left(1 + e^{-u(y)}\right)\right] \tag{4.3}$$

by dropping the condition $X = 1$.

Note that $u(y)$ is odd symmetric, i.e.,

$$u(-y) = -u(y). \tag{4.4}$$

We call u the *initial message* of the channel and call the distribution $p(u)$ of u the *initial density* of the channel, which will be used later for density evolution in Chapter 5. Richardson *et al.* showed that initial densities satisfy the following *symmetry condition*:

$$p(u) = e^u p(-u).$$

We give a brief proof of it here.

Lemma 4.2 *The initial density $p(u)$ of a symmetric channel satisfies the following*

symmetry condition*:*

$$u = \log \frac{p(u)}{p(-u)}. \tag{4.5}$$

Proof: The probability of u being in an infinitesimal interval du is $p(u)|du|$. In terms of $p(y|X=1)$, this is equal to the following:

$$p(u)|du| = \sum_{\{y_i|\ u=\log(p(y_i|X=1)/p(y_i|X=-1))\}} p(y_i|X=1)|dy_i|.$$

Using $u(-y) = -u(y)$ in (4.4), we get

$$\frac{p(u)|du|}{p(-u)|du|} = \frac{\sum_{\{y_i|\ u=\log(p(y_i|X=1)/p(y_i|X=-1))\}} p(y_i|X=1)|dy_i|}{\sum_{\{y_i|\ u=\log(p(y_i|X=1)/p(y_i|X=-1))\}} p(-y_i|X=1)|dy_i|},$$

where the set of y_i's is the same for the denominator and numerator. Since $u = \log(p(y_i|X=1)/p(y_i|X=-1)) = \log(p(y_i|X=1)/p(-y_i|X=1))$, we can replace $p(-y_i|X=1)$ in the denominator by $p(y_i|X=1)e^{-u}$. Therefore, we get

$$\log \frac{p(u)|du|}{p(-u)|du|} = \log \frac{1}{e^{-u}} = u.$$

□

4.2 Normalization of Symmetric Channels

For any discrete-input channel, the set of log likelihoods of the inputs given the channel output is a set of sufficient statistics for estimating the inputs. Since the log likelihoods are functions of the channel output, we may define a concatenated channel with the original channel followed by the set of log likelihood mappings. We call this the "normalized" channel of the original channel.[1]

This "normalization" of a channel is without loss of optimality for decoding since the output of the normalized channel is a set of sufficient statistics for the channel

[1]We cannot recover the original conditional output density in general after normalization.

input. This is also information lossless since sufficient statistics preserve mutual information [13]. Therefore, there is no loss in channel capacity after normalization.

In [53], Richardson *et al.* defined two symmetric channels *equivalent* if the initial densities are the same. They picked a symmetric channel with an output Y' and the transition probability $p(Y' = u|X = 1) = p(u)$ as a representative for each equivalence class by showing that the initial density of the representative channel is the same as that of the original channel. In fact, such a selection of a representative is equivalent to normalization.

In the case of binary-input symmetric-output channels, there are several more interesting properties of normalization. The following summarizes some of the properties of normalization.

1. A normalized channel is a symmetric channel.

2. The normalized channel of a normalized channel is itself.

3. The normalized channel of a symmetric channel has the same channel capacity as the original channel.

4. Normalized channel outputs are sufficient statistics for the input of the original channel. Therefore, decoding can be done using the normalized channel without loss of generality.

5. In designing low-density parity-check codes, we need to consider only the normalized channel without loss of generality.

6. Normalization simplifies some proofs involving various channels since the conditional output density of a normalized channel can be generated using a simple set of functions.

7. Iterative normalization in Section 5.2.3 becomes a motivation for developing a very accurate model of density evolution in Section 7.3.

8. Normalization unifies the output space to $\bar{\mathbb{R}}$, which is defined as $\mathbb{R} \cup \{-\infty, \infty\}$.

9. The conditional output density $p(u|X=1)$ of a normalized channel conditioned on $X=1$ satisfies the symmetry condition in (4.5).

10. The conditional output of the normalized channel of a clean channel (capacity $=1$), is always $\pm\infty$, which corresponds to the input ± 1, respectively.

11. The output of the normalized channel of a zero-capacity channel (capacity $=0$), is always zero.

12. Normalization removes degenerate cases such as duplicated output alphabets.

Now, we prove some of the above claims. We first prove that a normalized channel is a symmetric channel.

Lemma 4.3 *A normalized channel is a symmetric channel.*

Proof: Since $u(y)$ is odd-symmetric as given in (4.4), the concatenated channel consisting of a symmetric channel followed by $u(y)$ is a symmetric channel. □

Since the conditional output density $p(u|X=1)$ of a normalized channel is the initial density of the original channel, it is immediately clear that $p(u|X=1)$ satisfies the symmetry condition.

Property 2 follows directly from (4.5), which means that the LLR mapping of a normalized channel is an identity mapping. Using this observation, we can prove the following theorem.

Theorem 4.4 *The normalized channel of a symmetric channel has the same channel capacity as the original channel.*

Proof: First note that the channel capacity of both channels can be achieved using equi-probable input since they are symmetric channels. The channel capacity of a symmetric channel with transition probability $p(Y|X)$ is given by (4.3). Since the LLR mapping $\zeta(\cdot)$ of the normalized channel is an identity mapping, i.e., $\zeta(u(y)) = u(y)$, we get the channel capacity of the normalized channel as given in the following:

$$C \;=\; 1 - E\left[\log_2\left(1+e^{-u(y)}\right)\Big|\, X=1\right].$$

□

Richardson *et al.* defined the following set of basis functions and used it to prove useful theorems related to the probability of error for density evolution:

$$g_p(u) = \begin{cases} p\Delta_{-\log\frac{1-p}{p}}(u) + (1-p)\Delta_{\log\frac{1-p}{p}}(u), & \text{if } 0 < p \leq \frac{1}{2}; \\ \Delta_\infty(u), & \text{if } p = 0, \end{cases} \tag{4.6}$$

where $\Delta_x(u)$ is a delta function at $x \in \bar{\mathbb{R}}$ as used in [53]. Note that $g_p(u)$ is the initial density of a BSC with a cross-over probability p. We use the same basis to show that any normalized channel can be generated using the basis and thus we can effectively generate any symmetric channel using it.

We first decompose any initial density $p(u)$ into three parts, i.e.,

$$p(u) = \tilde{p}(u) + \varepsilon_0\Delta_0 + \varepsilon_\infty\Delta_\infty,$$

where $\tilde{p}(u)$ does not have any delta function at 0 or at ∞ and $\varepsilon_0, \varepsilon_\infty \geq 0$. Note that an initial density $p(u)$ cannot have a delta function at $-\infty$, since it violates the symmetry condition.

The following theorem shows how $p(u)$ can be generated from the basis.

Theorem 4.5 *Any initial density $p(u)$ can be generated using the basis $g_p(u)$ as follows:*

$$p(u) \;=\; \int_{\left[0,\frac{1}{2}\right]^+} g_p(u) f(p)\, dp, \tag{4.7}$$

where $\int_{\left[0,\frac{1}{2}\right]^+}$ denotes an integral from 0 to $\frac{1}{2}$ which fully integrates the delta functions at 0 and $\frac{1}{2}$ and $f(p)$ is given by

$$f(p) \;=\; \frac{1}{p(1-p)^2}\tilde{p}\left(\log\frac{1-p}{p}\right) + \varepsilon_0\Delta_{\frac{1}{2}}(p) + \varepsilon_\infty\Delta_0(p), \tag{4.8}$$

where we exclude $p = 0$ when evaluating the first term of (4.8).

Proof: The proof is straightforward. By using (4.8) in (4.7), we get

$$\int_{\left[0,\frac{1}{2}\right]^+} g_p(u) f(p)\, dp = \int_{\left(0,\frac{1}{2}\right)} g_p(u) \frac{1}{p(1-p)^2} \tilde{p}\left(\log \frac{1-p}{p}\right)\, dp + \varepsilon_0 \Delta_0 + \varepsilon_\infty \Delta_\varepsilon, \quad (4.9)$$

where $\int_{\left(0,\frac{1}{2}\right)}$ is an integral from 0 to $\frac{1}{2}$ that excludes the delta functions at 0 and $\frac{1}{2}$.

To evaluate the first term above, we define $x = \log \frac{1-p}{p}$ for $0 < p < \frac{1}{2}$. Using $dp = -p(1-p)dx$, we rewrite the integral as

$$\begin{aligned}
&\int_{\left(0,\frac{1}{2}\right)} g_p(u) \frac{1}{p(1-p)^2} \tilde{p}\left(\log \frac{1-p}{p}\right)\, dp \\
&\quad = \int_0^\infty \left[e^{-x}\delta(u+x) + \delta(u-x)\right] \tilde{p}(x)\, dx \\
&\quad = \begin{cases} \tilde{p}(u), & \text{if } u > 0; \\ e^u \tilde{p}(-u), & \text{if } u < 0. \end{cases}
\end{aligned}$$

Since $\tilde{p}(u)$ satisfies the symmetry condition, we get $e^u \tilde{p}(-u) = \tilde{p}(u)$. Therefore, by combining all three terms in (4.9), we get

$$p(u) = \tilde{p}(u) + \varepsilon_0 \Delta_0 + \varepsilon_\infty \Delta_\infty,$$

which completes the proof. □

Note that $f(p)$ satisfies the following three properties:

$$\begin{aligned}
f(p) &\geq 0; \\
\int_{\left[0,\frac{1}{2}\right]^+} f(p)\, dp &= 1; \\
f(p) &= 0, \quad \text{if } p < 0 \text{ or } p > \tfrac{1}{2}.
\end{aligned} \quad (4.10)$$

Therefore, $f(p)$ is a probability-like quantity. We prove the second property of the above in the following. The other two properties are obvious.

Lemma 4.6 *$f(p)$ in (4.8) satisfies $\int_{\left[0,\frac{1}{2}\right]^+} f(p)\, dp = 1$.*

Proof: As in Theorem 4.5, by using $x = \log \frac{1-p}{p}$, we get

$$\begin{aligned}\int_{\left[0,\frac{1}{2}\right]^+} f(p)\, dp &= \int_{\left[0,\frac{1}{2}\right]^+} \frac{1}{p(1-p)^2}\tilde{p}\left(\log\frac{1-p}{p}\right) + \varepsilon_0\Delta_{\frac{1}{2}}(p) + \varepsilon_\infty\Delta_0(p)\, dp \\ &= \varepsilon_0 + \varepsilon_\infty + \int_0^\infty \left(1+e^{-x}\right)\tilde{p}(x)\, dx.\end{aligned}$$

Since $\tilde{p}(-x) = e^{-x}\tilde{p}(x)$, the result follows. □

We show that the condition in (4.10) is enough for generating all densities that satisfy the symmetry condition.

Lemma 4.7 *Any $f(p)$ that satisfies (4.10) spans all densities that satisfy the symmetry condition.*

Proof: Since we have Theorem 4.5, we only have to prove that such $f(p)$ generates a density that satisfies the symmetry condition. This is effectively done in Theorem 4.5, since $f(p)$ in Theorem 4.5 is arbitrary except that it satisfies (4.10). □

Therefore, $f(p)$ spans all symmetric channels up to normalization. We call $f(p)$ of a symmetric channel the *compact initial density* of the channel.

The channel capacity in terms of the compact initial density $f(p)$ of a channel is given by the following:

$$\begin{aligned}C &= 1 - E\left[\log_2\left(1+e^{-u}\right)\right] \\ &= 1 - \int_{\bar{\mathbb{R}}}\int_{\left[0,\frac{1}{2}\right]^+} g_p(u)f(p)\, dp\ \log_2\left(1+e^{-u}\right)\ du \\ &= 1 - \int_{\left[0,\frac{1}{2}\right]^+} f(p)H(p)\, dp, \qquad (4.11)\end{aligned}$$

where $H(p) = -p\log_2 p - (1-p)\log_2(1-p)$ is the binary entropy function.

Using (4.11) and the monotonicity of $H(p)$, we conclude the following.

Lemma 4.8 *The initial density is always Δ_∞ if the channel is clean (capacity = 1) and is always Δ_0 if the channel capacity is zero.*

4.3 Summary

In this chapter, we have shown that by normalizing binary-input symmetric-output memoryless channels, we can create a unified approach to these channels. Several properties of normalization have been characterized. We have also shown that normalized channels can be generated by a simple set of basis functions.

Chapter 5

Low-Density Parity-Check Codes and Decoding

In this chapter, we introduce low-density parity-check (LDPC) codes, that were originally invented by Gallager in his thesis [26, 27], and some decoding algorithms for the codes.

We show how the sum-product algorithm and density evolution work. We also compare density evolution for the max-product algorithm to density evolution for the sum-product algorithm.

In this chapter, we also show how to implement density evolution algorithm efficiently, which we call *discretized density evolution*. We show that this implementation models the exact behavior of discretized sum-product decoding. Using this algorithm and an improved optimization algorithm, we design LDPC codes that approach the Shannon limit very closely in Chapter 8.

5.1 Codes on Graphs and Normal Realization

Gallager invented low-density parity-check codes in his thesis [26, 27], along with various iterative decoding algorithms including the sum-product algorithm. What would have been the most powerful codes were forgotten, however, because computing power was not enough at that time to demonstrate the full potential of the code.

Later, Tanner [63] replaced parity-checks in LDPC codes with general constraints to construct a more general class of codes known as Tanner codes.

By allowing hidden nodes in Tanner graphs, Wiberg [72, 73] made it possible to represent a code with states (such as a trellis code) as a graph, thus making a full connection between graphical and trellis representation of codes.

Recently, Wiberg's work was further generalized using "factor graphs" [39] to cover non-coding applications such as Markov random fields, belief networks and fast Fourier transforms.

Forney [19] constructed "normal graphs" by restricting symbols to be leaf edges connected to a constraint node and states to be ordinary edges between two constraint nodes. This form of restriction is without loss of generality in the sense that a factor graph can always be converted to a normal graph with the same topology and with about the same complexity when the sum-product algorithm is used. By normalizing a factor graph, the description of codes becomes more intuitive and easily visualized. More importantly, the fact that a dual graph of a normal realization represents a dual code is useful in understanding many interesting duality results.

In this thesis, we use a normal graph representation of LDPC codes as shown in Figure 5-1, which is a graph representation of the following $k \times n$ parity check matrix H:

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & \cdots & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & \cdots & 0 \\ & & & \vdots & & & & \vdots \\ 0 & 1 & 0 & 1 & 1 & 0 & \cdots & 0 \end{pmatrix}.$$

The codeword x is a binary column vector with n elements and satisfies the following parity check equation:

$$Hx = 0.$$

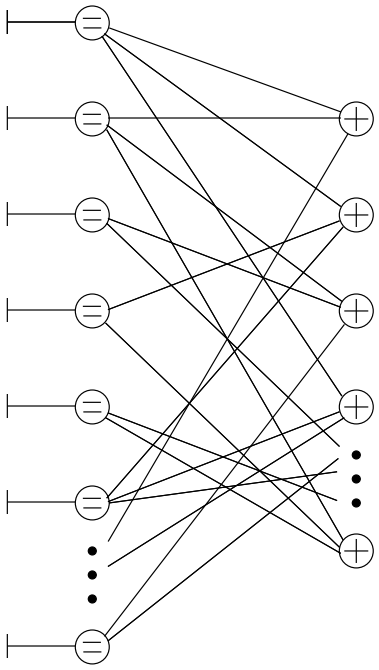


Figure 5-1: A normal graph representation of a low-density parity-check code, where leaf edges are symbols and every edge between an equality constraint on the left hand side and a parity check on the right hand side is a hidden state.

The term "low-density" means that the number of 1's in each row and column of H is small compared to the block length n.

We will sometimes call the repetition node in Figure 5-1 a variable since it represents the variable attached to it. In Figure 5-1, the number of parity checks attached to a equality constraint is called the degree of the variable that is attached to the equality constraint. Similarly we call the number of equality constraints attached to a parity check the degree of the check node.

We call the graph a regular LDPC code if all degrees are the same for variable nodes and also if all degrees are the same for check nodes, and an irregular LDPC code if not.

We assume symmetric channels defined in Section 4.1 for the rest of this thesis.

5.2 The Sum-Product Algorithm and Density Evolution

5.2.1 The Sum-Product Algorithm

Maximum-likelihood decoding usually becomes exponentially difficult for some graph codes including LDPC codes as the block length becomes large. Sum-product decoding, also known as belief propagation in the artificial intelligence community [50], can be viewed as applying Bayes' rule locally and iteratively to calculate approximate marginal *a posteriori* probabilities for these codes. The sum-product algorithm is also practical, since the decoding complexity per iteration is linear in block length. If a graph has no cycles, then it can be easily proved that the sum-product algorithm computes marginal posterior probabilities exactly.

However, in many situations of interest including LDPC code graphs as in Figure 5-1, we have graphs with cycles. In such cases, we may still want to run the sum-product algorithm ignoring cycles and hope for an answer that closely approximates the correct posterior probabilities. Despite a lack of theoretical understanding, the huge success of turbo codes and low-density parity-check codes has ignited further research in this area.

The behavior of the sum-product algorithm on graphs with a single cycle is relatively well understood [1, 20, 70]. In this case, the sum-product algorithm converges to a unique stationary point. If all variables are binary-valued, then the componentwise maximum likelihood estimates produced by running the sum-product algorithm are correct. However, it is hard to generalize this result to a graph with more than one cycle.

Another approach to understand the sum-product algorithm on a graph with many cycles is to assume that all variables are jointly Gaussian [23, 71, 57]. In this case, the analysis of the sum-product algorithm can be simplified since a Gaussian distribution is characterized by its mean and variance. A common result in [23, 71, 57] is that the sequence of means converges when certain conditions are satisfied. These papers also

show that (1) the means are correct when the sum-product algorithm converges and (2) the sequence of covariance matrices will converge to a unique fixed point.

We will describe the sum-product algorithm together with density evolution in more detail and explain how cycle problems can be avoided for LDPC codes in the next section.

5.2.2 Density Evolution

Richardson *et al.* [54, 53] demonstrated that the average asymptotic behavior of a sum-product decoder for LDPC codes is numerically computable by using an algorithm called *density evolution*. They also showed that for many interesting channels, including additive white Gaussian noise (AWGN) channels, one can calculate a *threshold* value for the ensemble of randomly constructed LDPC codes which determines the boundary of the error-free region asymptotically, as the block length tends to infinity.

We first describe how density evolution works for (d_v, d_c)-regular binary LDPC codes, where d_v denotes the number of neighbors of a variable node and d_c denotes the number of neighbors of a check node. Under the sum-product algorithm, variable and check nodes exchange messages iteratively.

A check node gets messages from its d_c neighbors, processes the messages, and sends the resulting messages back to its neighbors. Similarly, a variable node receives messages from its d_v neighbors and also from its corresponding channel output, processes the messages, and sends messages back to its neighbors.

Each output message of a variable or a check node is a function of all incoming messages to the node except for the incoming message on the edge where the output message will be sent out. This restriction is essential for the sum-product algorithm to produce correct marginal *a posteriori* probabilities for cycle-free graphs.

This two-step procedure is repeated many times. After n such iterations, the variable node decodes its associated bit based on all information obtained from its depth-n subgraph of neighbors. Under the "local tree assumption," namely that the girth of the graph is large enough so that the the subgraph forms a tree (i.e., there are

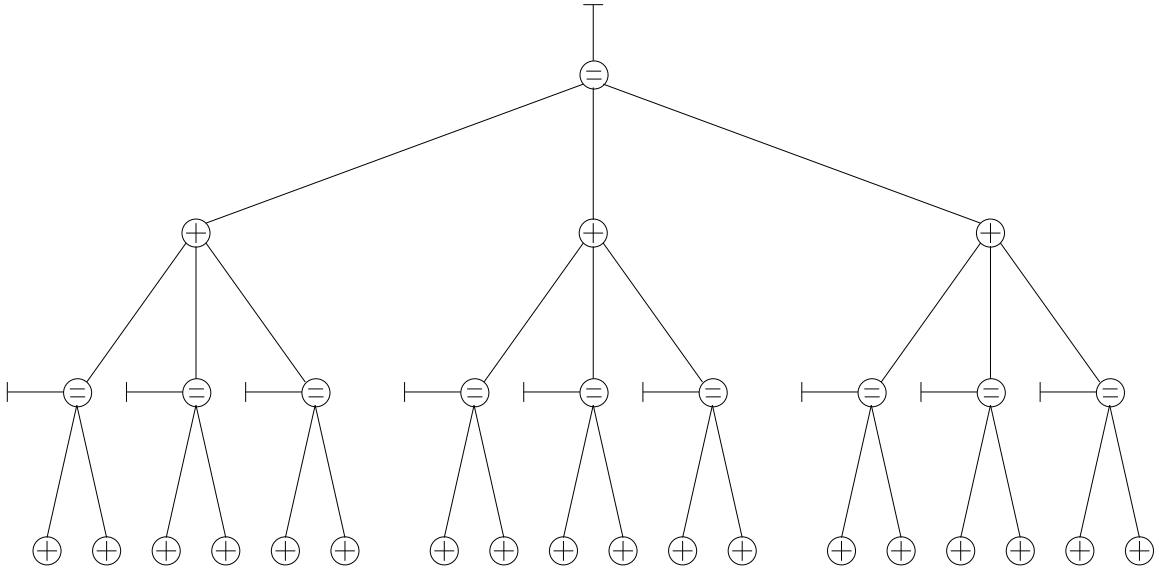


Figure 5-2: Sum-product decoding for a variable (root node)

no repeated nodes in the subgraph), we can analyze the decoding algorithm straightforwardly because incoming messages to every node are independent. Furthermore, by the general concentration theorem of [54], which generalizes the results of [44], we are assured that, for almost all randomly constructed codes and for almost all inputs, the decoder performance will be close to the decoder performance under the local tree assumption with high probability, if the block length of the code is long enough. From now on we base our analysis on this local tree assumption.

It is convenient to use log-likelihood ratios (LLRs) for messages; i.e., we use $v = \log \frac{p(y|x=1)}{p(y|x=-1)}$ as the output message of a variable node, where x is the bit value of the node and y denotes all the information available to the node up to the present iteration obtained from edges other than the one carrying v. Likewise, we define the output message of a check node as $u = \log \frac{p(y'|x'=1)}{p(y'|x'=-1)}$, where x' is the bit value of the variable node that gets the message from the check node, and y' denotes all the information available to the check node up to the present iteration obtained from edges other than the one carrying u.

Let v be a message from a variable node to a check node. Under sum-product

decoding, v is equal to the sum of all incoming LLRs; i.e.,

$$v = \sum_{i=0}^{d_v-1} u_i, \tag{5.1}$$

where $u_i, i = 1, \ldots, d_v - 1$, are the incoming LLRs from the neighbors of the variable node except for the check node that gets the message v, and u_0 is the initial message of the variable node as defined in Section 4.1. The density of the sum of d_v random variables $u_i, i = 0, \ldots, d_v - 1$, can be easily calculated by convolution of densities of the u_i's, which can be efficiently done in the Fourier domain [54]. Let $f_w(w)$ denote the density of a random variable w. Then, (5.1) becomes:

$$f_v = \bigotimes_{i=0}^{d_v-1} f_{u_i},$$

where $\bigotimes$ denotes convolution.

The message update rule for check nodes can be obtained from the duality between variable and check nodes and the resulting Fourier transform relationship between (p, q) and $(p + q, p - q)$, where $p = p(x = 1|y)$ and $q = p(x = -1|y)$ [19]. From this, we get the following "tanh rule" [35, 3, 34, 54, 19]:

$$\tanh \frac{u}{2} = \prod_{j=1}^{d_c-1} \tanh \frac{v_j}{2}, \tag{5.2}$$

where $v_j, i = 1, \ldots, d_c - 1$, are the incoming LLRs from $d_c - 1$ neighbors of a check node, and u is the message sent to the remaining neighbor.

The output u can be written simply as

$$u = \mathcal{R}_{d_c-1}(v_1, v_2, \ldots, v_{d_c-1}),$$

where $\mathcal{R}_n : \mathbb{R}^n \to \mathbb{R}$ is defined as

$$\mathcal{R}_n(v_1, v_2, \ldots, v_n) = 2 \tanh^{-1} \left(\prod_{i=1}^{n} \tanh \frac{v_i}{2} \right). \tag{5.3}$$

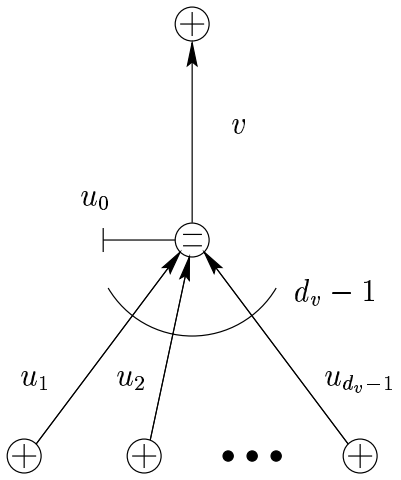


Figure 5-3: Message flow through a variable node

Figures 5-3 and 5-4 show message flows through a variable and a check node, respectively, where we use normal realizations [19] of variable and check nodes, which become a repetition and a parity check, respectively. In this representation, half-edges represent variables and full edges represent states. Under sum-product decoding, constraints become computation nodes and states become communication links between constraint nodes. Figure 5-4 (b) shows how the decoding for a check node can be done on a dual graph (repetition code) using Fourier transforms, which is a special case of a more general idea of dualizing the sum-product algorithm on a dual graph [19].

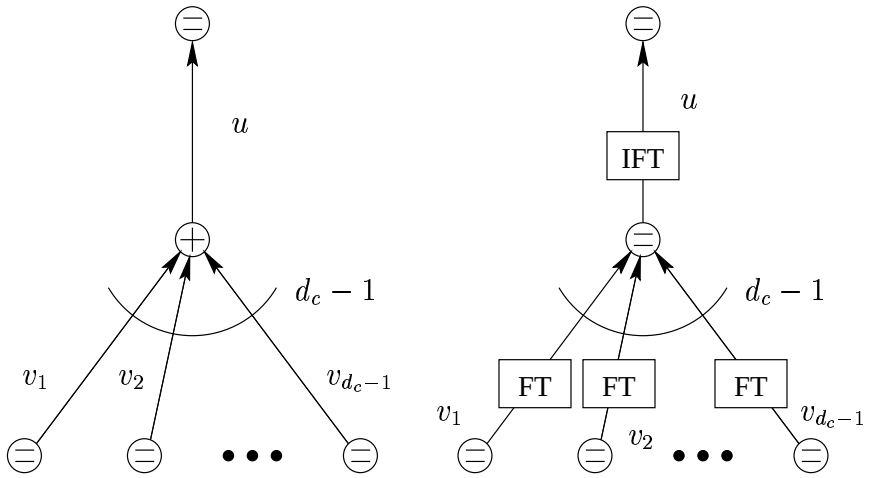
To apply the same Fourier transform technique to (5.2), we need to take logarithms on each side to convert the product into a sum, i.e.,

$$\left(s_u, \log\left|\tanh\frac{u}{2}\right|\right) \;=\; \sum_{j=1}^{d_c-1}\left(s_{v_j}, \log\left|\tanh\frac{v_j}{2}\right|\right), \tag{5.4}$$

where s_u and s_{v_j} are the signs of u and v_j, $1 \le j \le d_c - 1$, respectively. We define the sign s_x as 0 if $x \ge 0$ and 1 otherwise. Thus, the sum for the sign in (5.4) is performed in $\mathbb{Z}_2$ and the sum for the magnitude is the ordinary sum in $\mathbb{R}$. As in the first case, density evolution for this step can be done numerically in the Fourier domain [54].[1].

This can be generalized to irregular codes, where a variable number of degrees is

[1]Density evolution can be described more rigorously by using cumulative distribution functions as was done in [53]



(a) Original (b) Equivalent message flow

Figure 5-4: Message flow through a check node

allowed. Let $\lambda(x) = \sum_{i=2}^{d_l} \lambda_i x^{i-1}$ and $\rho(x) = \sum_{i=2}^{d_r} \rho_i x^{i-1}$ be the generating functions of the degree distributions for the variable and check nodes, respectively, where λ_i and ρ_i are the fractions of edges belonging to degree-i variable and check nodes, respectively [45]. Using this expression, the nominal rate r of the code is given by $r = 1 - \frac{\int_0^1 \rho(x)}{\int_0^1 \lambda(x)}$ [45]. We will describe density evolution for irregular cases in the following section.

This two-stage computation, called density evolution, makes it feasible to calculate the thresholds of message-passing decoding in [54]. First, without loss of generality, we assume that the all-0 codeword was sent. Then, we fix the channel parameter, namely noise power, and we run the above algorithm iteratively until either the density of v tends to the Δ_∞ (equivalently, the probability of error tends to zero), or it converges to a density with a finite probability of error (the probability of v being negative).[2] The threshold is defined as the maximum noise level such that the probability of error tends to zero as the number of iterations tends to infinity.

Richardson *et al.* proved that during the density evolution for the sum-product algorithm, the symmetry condition (4.5) is preserved for all messages if the initial

[2] Δ_0 contributes half of its probability to the probability of error.

message satisfies the condition [53]. They also showed that if the symmetry condition is preserved in all messages, then the following are satisfied.

1. If the probability of error of a message is 0, then the distribution is Δ_∞.

2. The probability of error at the ℓ-th iteration is a non-increasing function of ℓ.

We will show in Section 5.3 that for density evolution for the max-product algorithm, the symmetry condition is not preserved.

Richardson *et al.* also showed that there exists a condition called *stability condition* that controls the behavior of density evolution near the zero probability of error. Let N be the channel parameter of a symmetric channel. Then, if $\lambda'(0)\rho'(1) < S(N)^{-1}$, then the error probability will converge to zero if it is initially small enough, and if $\lambda'(0)\rho'(1) > S(N)^{-1}$, then the error probability is bounded away from zero [53, 55], which is a generalization of Shokrollahi's result for BECs [61]. $S(N)$ is the *stability function*[3] of the channel and is given by the following [53]:

$$S(N) = \int_{\mathbb{R}} e^{-u/2} p(u)\, du, \tag{5.5}$$

where $p(u)$ is the initial density of the channel.

Using the compact initial density $f(p)$ of the channel, we can simply describe the stability function as follows:

$$\begin{aligned} S(N) &= \int_{\mathbb{R}} \int_{\left[0,\frac{1}{2}\right]^+} g_p(u) f(p) e^{-u/2}\, dp\, du \\ &= \int_{\left[0,\frac{1}{2}\right]^+} f(p) 2\sqrt{p(1-p)}\, dp \end{aligned} \tag{5.6}$$

In Section 6.1, we will provide channel examples and their stability functions.

[3]This is the same as the constant in the Bhattacharyya bound [66] for symmetric channels.

5.2.3 Iterative Normalization

In Figure 5-3, the output message v can be interpreted as the channel output of a channel whose input is the value of the hidden state carrying the message v and whose output is the set of all observations made in the subtree below the hidden state. Since this output message is in LLR format, this equivalent channel has already been normalized.

Similarly, in Figure 5-4, the output message u can be viewed as the channel output of a channel whose input is the value of the hidden state carrying the message u and whose output is the set of all observations made in the subtree below the hidden state. This channel is also normalized.

Therefore, density evolution can be interpreted as an iterative normalization, i.e., at each variable or check node we form an equivalent normalized channel as described above and these steps are repeated. When this iteration is completed at the root node, there will be one normalized channel.

Using the same proof as the correctness of the sum-product algorithm for a cycle-free graph, we can prove that this iterative normalization is equivalent to the normalization at once, i.e., we normalize the channel whose input is the value of the root node and its output is the set of all observations made in the subtree of the root node.

Using the same argument used in density evolution, we can conclude that, in the limit, there are two possibilities in the normalized channel. If the noise is less than the threshold of the code, then the normalized channel is a clean channel with capacity = 1. If the noise is greater than the threshold, then the normalized channel is a noisy channel with a nonzero probability of error.

This concept will be used to develop an accurate one-dimensional approximation method, namely a Gaussian-capacity approximation, in Section 7.3.

5.2.4 Discretized Density Evolution

From now on, we assume a random ensemble of irregular codes specified by two degree distributions $\lambda(x)$ and $\rho(x)$, where $\lambda(x) = \sum_{i=2}^{d_l} \lambda_i x^{i-1}$ and $\rho(x) = \sum_{j=2}^{d_r} \rho_j x^{j-1}$. Here,

λ_i is the fraction of edges that belong to degree-i variable nodes, ρ_j is the fraction of edges that belong to degree-j check nodes, d_l is the maximum variable degree, and d_r is the maximum check degree.

To perform density evolution numerically, we discretize messages in the following way. Let $\mathcal{Q}(w)$ be the quantized message of w, i.e.,

$$\mathcal{Q}(w) = \begin{cases} \lfloor \frac{w}{\Delta} + \frac{1}{2} \rfloor \cdot \Delta, & \text{if } w \geq \frac{\Delta}{2}; \\ \lceil \frac{w}{\Delta} - \frac{1}{2} \rceil \cdot \Delta, & \text{if } w \leq -\frac{\Delta}{2}; \\ 0, & \text{otherwise,} \end{cases} \tag{5.7}$$

where $\mathcal{Q}$ is the quantization operator, Δ is the quantization interval, $\lfloor x \rfloor$ is the largest integer not greater than x, and $\lceil x \rceil$ is the smallest integer not less than x. Note that the definition of $\mathcal{Q}$ is symmetrical with respect to sign, i.e., $\mathcal{Q}(-x) = -\mathcal{Q}(x)$.

Discretized sum-product decoding is defined as sum-product decoding with all input and output messages quantized in this way. Under discretized sum-product decoding, (5.1) becomes $\bar{v} = \sum_{i=0}^{d_v - 1} \bar{u}_i$, where $\bar{v} = \mathcal{Q}(v)$ and $\bar{u}_i = \mathcal{Q}(u_i)$ for $i = 0, \ldots, d_v - 1$. We denote the probability mass function (pmf) of a quantized message $\bar{w}$ by $p_{\bar{w}}[k] = \Pr(\bar{w} = k\Delta)$ for $k \in \mathbb{Z}$. Then, $p_{\bar{v}}$ is related to its input pmf's by

$$p_{\bar{v}} = \bigotimes_{i=0}^{d_v - 1} p_{\bar{u}_i},$$

where $p_{\bar{v}}$ is the pmf of $\bar{v}$, $p_{\bar{u}_i}$ is the pmf of $\bar{u}_i$, and $\bigotimes$ is discrete convolution. Since the $\bar{u}_i$'s are independent and identically distributed (iid) for $1 \leq i < d_v$, the above can be rewritten as

$$p_{\bar{v}} = p_{\bar{u}_0} * \left(\bigotimes^{d_v - 1} p_{\bar{u}} \right), \tag{5.8}$$

where $p_{\bar{u}} = p_{\bar{u}_i}$, $1 \leq i < d_v$. This calculation can be done efficiently using an FFT.

We define the following two-input operator $\mathcal{R}$:

$$\mathcal{R}(a, b) = \mathcal{Q}\left(\mathcal{R}_2(a, b) \right),$$

where a and b are quantized messages and $\mathcal{R}_2$ is defined in (5.3). Note that this operation can be done using a pre-computed table, which is the key step for making discretized density evolution computationally efficient. Using this operator, we calculate the quantized message $\bar{u}$ of (5.2) as follows:

$$\bar{u} = \mathcal{R}(\bar{v}_1, \mathcal{R}(\bar{v}_2, \dots, \mathcal{R}(\bar{v}_{d_c-2}, \bar{v}_{d_c-1}) \dots)),$$

where we assume that discretized sum-product decoding at check nodes is done pairwise.

Let $c = \mathcal{R}(a, b)$. The pmf p_c of c is given by

$$p_c[k] = \sum_{(i,j):k\Delta=\mathcal{R}(i\Delta,j\Delta)} p_a[i]p_b[j].$$

Abusing notation, we write this as $p_c = \mathcal{R}(p_a, p_b)$.

Since the $p_{\bar{v}_i}$'s are all equal, we define $p_{\bar{v}} = p_{\bar{v}_i}$ for any $1 \le i < d_c$, and write $p_{\bar{u}} = \mathcal{R}(p_{\bar{v}}, \mathcal{R}(p_{\bar{v}}, \dots, \mathcal{R}(p_{\bar{v}}, p_{\bar{v}}) \dots))$ as $p_{\bar{u}} = \mathcal{R}^{d_c-1}p_{\bar{v}}$. Note that we can reduce the number of operations to do this by properly nesting this calculation. Although different nesting might produce slightly different results due to finite quantization, it still corresponds to some valid discretized sum-product decoding, and asymptotically, as $\Delta \to 0$, it will converge to continuous belief propagation.

For irregular codes, this nesting-based computation is even more efficient, since we can reuse output pmf's of lower degree nodes and also intermediate pmf's. For example, the output pmf of a degree-9 check node can be computed in three steps (convolution of two input pmf's, convolution of two of the output pmf's of the previous convolution, and one more convolution to combine two pmf's from the second convolution) and the output density of a degree-13 check node can be done in just one step using the output of the previous second convolution and the output pmf of the degree-9 node.

In fact, we use nesting for variable nodes, too. Since the discrete convolution in (5.8) is actually a circular convolution, we need to have some zero-padding to prevent

aliasing. If we compute the convolution pairwise, then we only need to pad the same number of zeros as the array size of the pmf. This becomes efficient especially for irregular codes with a large maximum variable degree.

By defining $\lambda(p) = \sum_{i=2}^{d_l} \lambda_i \bigotimes^{i-1} p$ and $\rho(p) = \sum_{j=2}^{d_r} \rho_j \mathcal{R}^{j-1} p$, we can write discretized density evolution as follows:

Theorem 5.1 *Discretized density evolution is described by*

$$p_{\bar{u}}^{(\ell+1)} = \rho(p_{\bar{u}_0} * \lambda(p_{\bar{u}}^{(\ell)})),$$

where the initial pmf is $p_{\bar{u}}^{(0)} = \Delta_0$ *and* ℓ *is the iteration number.*

The complexity of this calculation is of order $O(n^2)$ due to the calculations at check nodes, where n is the number of quantization levels. However, this is actually faster than the calculation based on changing of measures between LLR and $(p_0 - p_1)$ domains as in [53], which requires finer quantization due to numerical problems when changing measures. As a result, our algorithm is more accurate and also realizes the discretized version of the sum-product algorithm exactly. This implies that the threshold predicted by our algorithm is always a lower bound, since it is exact for a sub-optimal decoding [54].

Using this discretized density evolution algorithm, we can easily calculate threshold values with a precision of six digits or more. Table 5.1 shows how the threshold values are affected by quantization for a (3,6) regular code for the AWGN channel. When the range of LLR $= [-25, 25]$ is used, it shows that a 14-bit quantization is enough for 6-digit precision. It also shows that the precision depends only on the quantization step size, not the range of LLR if the maximum LLR is at least 25. In fact, we have observed that we can decrease the range of LLR further to about 20 without sacrificing precision. We have observed that for some irregular codes the probability of error cannot be decreased below a certain number, which increases as we decrease the maximum LLR. For maximum LLR $= 25$, this number is very small, less than 10^{-6} from our observations. Since this number is so small, it seems we can rely on the stability condition to predict if the error probability will eventually con-

Table 5.1: Quantization effect for a (3,6) code. $[-25, 25]$ and $[-50, 50]$ are used for the ranges of LLR. Number of bits used for quantization versus threshold values in σ and errors in dB relative to the threshold value for 14-bit quantization and for maximum LLR = 25. All thresholds are rounded down.

	LLR $= [-25, 25]$		LLR $= [-50, 50]$	
bits	threshold (σ)	error [dB]	threshold (σ)	error [dB]
9	0.880616	0.00289		
10	0.880832	0.00077	0.880616	0.00289
11	0.880890	0.00020	0.880832	0.00077
12	0.880905	0.00005	0.880890	0.00020
13	0.880909	0.00001	0.880905	0.00005
14	0.880910	0.00000	0.880909	0.00001

Table 5.2: Quantization effect for the $d_l = 200$ code in Table 8.3. Number of bits used for quantization versus threshold values in σ and errors in dB relative to the threshold value for 14-bit quantization. Maximum LLR = 25 was used. All thresholds are rounded down.

bits	threshold (σ)	error [dB]
9	0.975122	0.01708
10	0.976918	0.00109
11	0.977025	0.00014
12	0.977037	0.00003
13	0.977040	0.00001
14	0.977041	0.00000

verge to zero, although we do not have a rigorous quantitative analysis. In practical implementations of decoders, this should not be a problem since block lengths are typically very small compared to the inverse of the number.

Table 5.2 shows how the threshold values are affected by quantization for the rate-1/2 $d_l = 200$ code in Table 8.3. When 14 bits are used to quantize the LLR values into 16,384 levels, we observe that the threshold has 6-digit precision as in the (3,6) case.

Based on the technique developed in this section, we will show how to optimize degree distributions in Chapter 8.

5.3 Density Evolution for the Max-Product Algorithm

In this section, we analyze density evolution for the max-product algorithm, which is equivalent to the min-sum algorithm except that costs are probabilities rather than negative logs of probabilities. Unfortunately, it turns out that the max-sum algorithm does not preserve the "symmetry condition." Therefore, it is not guaranteed that the probability of error is non-increasing. In fact, we show an example where the probability of error actually increases as iterations go on.

On a cycle-free graph, decoding based on the max-product algorithm produces a probability of error for each bit that is always greater than or equal to that of the sum-product algorithm, because the sum-product algorithm produces the minimum probability of error.

5.3.1 The Max-Product Algorithm

In the max-product algorithm, for each value of the output state, the most probable valid input combination is chosen, whereas in the sum-product algorithm, "maximum" is replaced by "sum." For variable nodes, "maximum" is equal to "sum" because there is only one possible combination of inputs for each output value. As in the sum-product case, we use LLRs as messages for the max-product algorithm. Therefore, the output LLR v of a variable node has the same form as in the sum-product algorithm case, as follows:

$$v = u_0 + \sum_{i=1}^{d_v - 1} u_i,$$

where u_0 is the channel message, u_i's are incoming LLR messages from edges other than the one carrying v, and d_v is the degree of the node.

For a degree-3 check node, the output probabilities (p_0, p_1) are given by the fol-

lowing:

$$\begin{aligned} p_0 &= \alpha \max(p_0^{(1)} p_0^{(2)}, p_1^{(1)} p_1^{(2)}) \\ &\text{and} \\ p_1 &= \alpha \max(p_0^{(1)} p_1^{(2)}, p_1^{(1)} p_0^{(2)}), \end{aligned} \tag{5.9}$$

where α is a normalization constant and $(p_0^{(i)}, p_1^{(i)})$ are i-th input probabilities. In terms of LLRs, (5.9) becomes the following:

$$\begin{aligned} u &= \begin{cases} v_2, & \text{if } v_1 \geq |v_2|; \\ -v_2, & \text{if } v_1 \leq -|v_2|; \\ v_1, & \text{if } v_2 \geq |v_1|; \\ -v_1, & \text{if } v_2 \leq -|v_1|; \end{cases} \\ &= \min(|v_1|, |v_2|) \operatorname{sgn}(v_1) \operatorname{sgn}(v_2), \end{aligned}$$

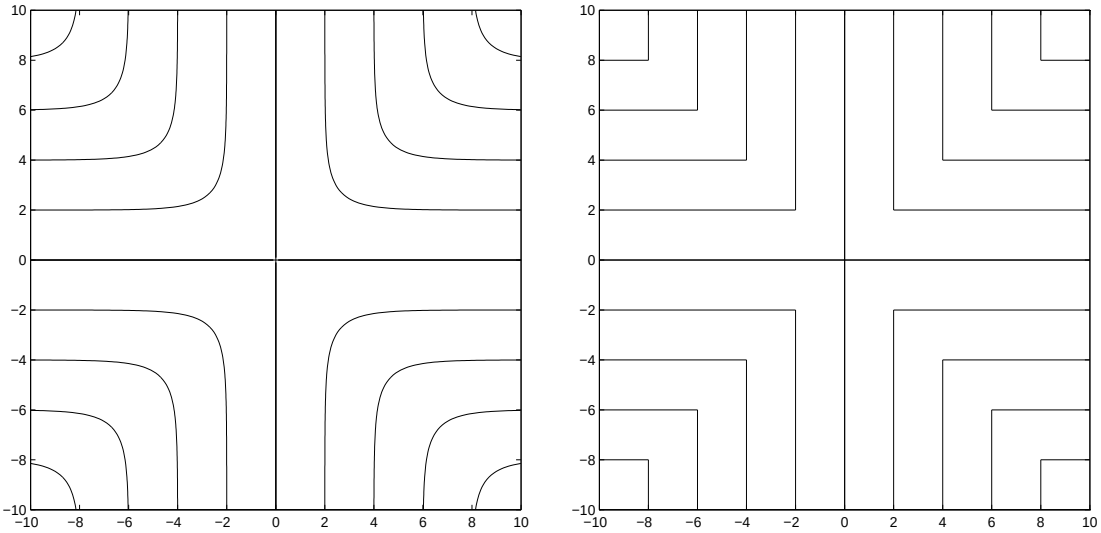
where u is the output LLR, v_i is the input LLRs from edges other than the one carrying u, and $\operatorname{sgn} x = 1$ if $x \geq 0$ and $\operatorname{sgn} x = -1$ otherwise.

Figure 5-5 shows contour plots of the output message u of a degree-3 check node for both sum-product and max-product cases, where u is given by $2 \tanh^{-1} \left(\tanh \frac{v_1}{2} \tanh \frac{v_2}{2} \right)$ for the sum-product case. Observe that as $|v_1|$ or $|v_2|$ tends to infinity, u becomes the same for these two algorithms.

Since the check-sum can be computed pairwise iteratively, the output u of a degree d_c check node using the max-product algorithm becomes the following regardless of the order of the calculations, which is due to Wiberg [72].

Lemma 5.2 *For a degree-d_c check node, its LLR output u becomes*

$$u = \mathcal{M}_{d_c - 1}(v_1, v_2, \ldots, v_{d_c - 1}),$$



(a) Sum-product

(b) Max-product

Figure 5-5: Contour plots of the output message of a degree-3 check node from (a) sum-product and (b) max-product algorithms

where $\mathcal{M}_n : \mathbb{R}^n \to \mathbb{R}$ *is defined as*

$$\mathcal{M}_n(v_1, v_2, \ldots, v_n) = \min(|v_1|, |v_2|, \ldots, |v_n|) \prod_{i=1}^{n} \operatorname{sgn}(v_i),$$

where the v_i*'s are input LLRs from edges other than the one carrying* u*.*

When we implement the max-product algorithm for a degree-d_c check node, we can efficiently calculate all d_c output messages by first storing the minimum absolute value m_1 and the next smallest absolute value m_2 of the d_c incoming messages and also by storing the parity of the d_c incoming messages. Note that m_1 can be equal to m_2 in general. The parity of each output message is determined by the mod-2 sum of the overall parity and the parity of the input message on the edge that carries the output message. The absolute value of the output message is equal to m_2 if the absolute value of the incoming message on the edge that carries the output message is equal to m_1, and is equal to m_1 otherwise.

5.3.2 Equivalence of Sum-Product and Max-Product Algorithms

We note that at coarse scales, the two contour plots of Figure 5-5 become similar. In fact, when we discretize the sum-product algorithm in a certain way, it becomes equivalent to the max-product algorithm if and only if the quantization step size Δ is greater than or equal to a certain number Δ^*, if the max-product algorithm is discretized in the same way (which we call the "discretized max-product algorithm").

Since the sum-product algorithm is the same as the max-product algorithm for calculations at variable nodes, we only need to consider the calculations at check nodes. We assume that the output u of a degree-d_c check node is given by the following:

$$u = \mathcal{Q}\left(\mathcal{R}_{d_c-1}(\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_{d_c-1})\right), \tag{5.10}$$

where the $\bar{v}_i = \mathcal{Q}(v_i), 1 \leq i \leq d_c - 1$, are quantized inputs.

Without loss of generality, we have to consider only the case when all inputs are nonnegative, since the quantization operator $\mathcal{Q}$ in (5.7) is defined symmetrically with respect to sign ($\mathcal{Q}(-x) = -\mathcal{Q}(x)$) and since $\mathcal{M}_n$ and $\mathcal{R}_n$ are parity-preserving (Definition 5.1).

Furthermore, since $\mathcal{M}_n$ and $\mathcal{R}_n$ are symmetric, we need to consider only the case when $0 \leq v_1 \leq v_2 \leq \ldots \leq v_n$. In this case, $\mathcal{M}_n(v_1, v_2, \ldots, v_n) = v_1$. Now the question is: what is the quantization step size Δ that makes $\mathcal{Q}(\mathcal{R}_n(\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_n))$ equal to $\bar{v}_1$ for all $0 \leq v_1 \leq v_2 \leq \ldots \leq v_n$? We first prove the following lemma.

Lemma 5.3 *$\mathcal{R}_n(v_1, v_2, \ldots, v_n) \leq v_1$ for all $0 \leq v_1 \leq v_2 \leq \ldots \leq v_n$.*

Proof: By taking a monotonically increasing function $\tanh\frac{(\cdot)}{2}$ on both sides, we get the result since $\tanh\frac{v_i}{2} < 1$ for $2 \leq i \leq n$. $\square$

If the last m inputs are the same and equal to v, i.e., $v_{n-m+1} = \cdots = v_n = v$, then $\mathcal{R}_n(v_1, v_2, \ldots, v_{n-m}, v, \ldots, v)$ is a monotonically increasing function of v. Therefore

we have to consider only the case when $v = v_{n-m}$, which gives the biggest difference between $\mathcal{Q}(\mathcal{R}_n(\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_{n-m}, v, \ldots, v))$ and $\bar{v}_1$. By finite induction, we conclude that we have to consider only the case where all inputs are the same, i.e., $v_1 = v_2 = \cdots = v_n = v$. From this, we prove the following theorem.

Theorem 5.4 *$\mathcal{Q}(\mathcal{R}_n(\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_n))$ is equal to $\bar{v}_1$ for all $0 \leq v_1 \leq v_2 \leq \ldots \leq v_n$ iff $\Delta \geq \Delta^*$, where $\Delta^* = 2 \ln n$.*

Proof: Since $v - \mathcal{R}_n(v, v, \ldots, v)$ is a monotonically increasing function of v, we have to consider only the case when $v \to \infty$, where in the limit we obtain the following value:

$$\lim_{v \to \infty} v - \mathcal{R}_n(v, v, \ldots, v) = \ln n.$$

Therefore, $\mathcal{Q}(\mathcal{R}_n(\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_n))$ is equal to $\bar{v}_1$ for all $0 \leq v_1 \leq v_2 \leq \ldots \leq v_n$ iff $\bar{v} - \ln n \geq \bar{v} - \frac{\Delta}{2}$. $\square$

For an irregular code with maximum check degree d_r, this means that the discretized sum-product algorithm given by (5.10) is equivalent to the discretized max-product algorithm iff $\Delta \geq 2 \ln(d_r - 1)$. The discretized sum-product algorithm in Section 5.2.4, which is more practical than (5.10) since the decoding can be done efficiently using a small table, is equivalent to the discretized max-product algorithm iff $\Delta \geq 2 \ln 2 \sim 1.386$, because $n = 2$ in this case. In this case, if the range of LLR is from -25 to 25, which is enough for reasonably good accuracy in calculating thresholds, we conclude that if we have fewer than 6 bits for quantizing messages, then the discretized sum-product algorithm becomes equivalent to the discretized max-product algorithm.

One implication is that if we discretize sum-product decoding, then it does not preserve the symmetry condition in general. In fact, we have observed that in some cases the probability of error increases as iterations go on, especially when the quantization step size is not small.

5.3.3 Density Evolution for the Max-Product Algorithm

Density evolution for the max-product algorithm is simpler to describe than density evolution for the sum-product algorithm, due to its simpler decoding rule. Since the max-product algorithm is equivalent to the sum-product algorithm at variable nodes, we describe density evolution only at check nodes. Let f_{v_i} be the pdf of the input message v_i to a degree-d_c check node, where $1 \leq i \leq d_c$. Let F_{v_i} be the cumulative distribution function (cdf) of v_i, i.e., $F_{v_i}(x) = \Pr(v_i \leq x)$.

The cdf $F_u(x)$ of the d_c-th output u of the check node is given by the following when $x \geq 0$:

$$\begin{aligned} F_u(x) &= 1 - \Pr\left(\min(v_1, v_2, \ldots, v_{d_c-1}) \prod_{i=1}^{d_c-1} \operatorname{sgn} v_i > x \geq 0\right) \\ &= 1 - \sum_{s_j=\pm 1, 1 \leq j \leq d_c-1: \prod s_j = 1} \prod_{i=1}^{d_c-1} \Pr(|v_i| > x, \operatorname{sgn} v_i = s_i), \end{aligned}$$

where the s_i's are ± 1 for $1 \leq i \leq d_c - 1$. $\Pr(|v_i| > x, \operatorname{sgn} v_i = s_i)$ can be written simply as the following:

$$\Pr(|v_i| > x, \operatorname{sgn} v_i = s_i) = \begin{cases} 1 - F_{v_i}(x), & \text{if } s_i = 1; \\ F_{v_i}(-x-0), & \text{if } s_i = -1, \end{cases}$$

where $F_{v_i}(-x-0)$ denotes $\lim_{t \to (-x)^-} F_{v_i}(t)$. Using this and the fact that the v_i's are iid, we get the following:

$$\begin{aligned} F_u(x) &= 1 - \sum_{k=0:\, k=\text{even}}^{d_c-1} \binom{d_c-1}{k} (F_v(-x-0))^k (1 - F_v(x))^{d_c-1-k} \\ &= 1 - \frac{1}{2}(1 - F_v(x) + F_v(-x-0))^{d_c-1} \\ &\quad - \frac{1}{2}(1 - F_v(x) - F_v(-x-0))^{d_c-1}, \end{aligned} \tag{5.11}$$

where F_v is equal to F_{v_i} for all $1 \leq i \leq d_c$.

When $x < 0$, the cdf $F_u(x)$ of the d_c-th output u of the check node can be

calculated as the following:

$$\begin{aligned} F_u(x) &= \Pr\left(\min(v_1, v_2, \ldots, v_{d_c-1}) \prod_{i=1}^{d_c-1} \operatorname{sgn} v_i \leq x < 0\right) \\ &= \sum_{s_j=\pm 1, 1\leq j\leq d_c-1:\ \prod s_j=-1} \prod_{i=1}^{d_c-1} \Pr(|v_i| \geq -x, \operatorname{sgn} v_i = s_i), \end{aligned}$$

where s_i's are ± 1 for $1 \leq i \leq d_c - 1$. Similarly as before, $\Pr(|v_i| \geq -x, \operatorname{sgn} v_i = s_i)$ can be written as the following:

$$\Pr(|v_i| \geq -x, \operatorname{sgn} v_i = s_i) = \begin{cases} 1 - F_{v_i}(-x - 0), & \text{if } s_i = 1; \\ F_{v_i}(x), & \text{if } s_i = -1. \end{cases}$$

Using this and the fact that the v_i's are iid as before, we get the following:

$$\begin{aligned} F_u(x) &= \sum_{k=1:\ k=\text{odd}}^{d_c-1} \binom{d_c - 1}{k} (F_v(x))^k (1 - F_v(-x - 0))^{d_c-1-k} \\ &= \frac{1}{2}(1 - F_v(-x - 0) + F_v(x))^{d_c-1} \\ &\quad -\frac{1}{2}(1 - F_v(-x - 0) - F_v(x))^{d_c-1}. \end{aligned} \tag{5.12}$$

The pdf $f_u(x)$ is obtained by differentiating $F_u(x)$ in (5.11) and (5.12).

5.3.4 The Symmetry Condition

Although the two algorithms are equivalent at coarse scales, we note that the max-product algorithm does not have the key desirable property of preserving the symmetry condition. We show this by using the basis functions $g_p(x)$ in (4.6).

Lemma 5.5 *If the input distributions are $g_p(x)$, $0 < p < \frac{1}{2}$, then the output distribution of a degree-3 check node does not satisfy the symmetry condition when the max-product algorithm is used.*

Proof: The output distribution $p(x)$ is

$$p(x) = 2p(1-p)\delta\left(x + \log\frac{1-p}{p}\right) + (p^2 + (1-p)^2)\delta\left(x - \log\frac{1-p}{p}\right),$$

which does not satisfy the symmetry condition. □

Therefore, there is no guarantee that the probability of error is non-increasing for the max-product algorithm, which needs to hold for decoding algorithms that preserve the symmetry condition. In fact, we show that it can increase in some cases in the following example.

Example 5.1 *We consider a (3,4)-regular LDPC code for the BSC with crossover probability p. The initial distribution is given by g_p. After one iteration, the output distribution of a check node becomes*

$$q\delta\left(x + \log\frac{1-p}{p}\right) + (1-q)\delta\left(x - \log\frac{1-p}{p}\right),$$

where $q = p^3 + 3p(1-p)^2$. The probability of error of any variable after one iteration becomes

$$pq^3 + 3pq^2(1-q) + q^3(1-p) + \frac{3}{2}pq(1-q)^2 + \frac{3}{2}q^2(1-p)(1-q).$$

Figure 5-6 shows this probability of error as a function of p. We observe that if p is greater than about 0.0845, then the probability of error increases after one iteration. (The Shannon limit is about 0.2145, the threshold for the sum-product algorithm is about 0.1670, and the threshold for the max-product algorithm is about 0.1229.)

Another interesting phenomenon observed in Figure 5-6 is that if p is less than about 0.1226, then the probability of error after two iterations is smaller than the probability of error after one iteration. This means that if p is between about 0.0845 and 0.1226, then the probability of error increases initially and then decreases.

If p is between about 0.1227 and 0.1229, then the probability of error increases for at least two iterations initially, but converges to zero eventually. Figure 5-7 shows

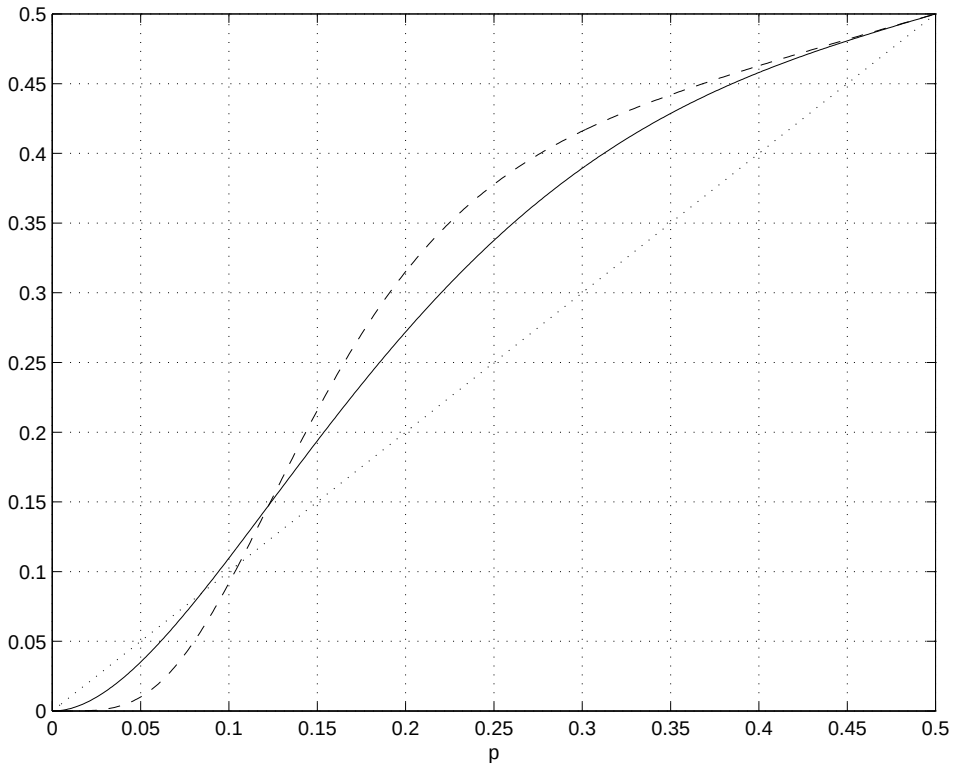


Figure 5-6: The probability of error after one iteration (—) and after two iterations (- -) as functions of the initial probability of error p for a (3,4)-regular LDPC code for the BSC using the max-product algorithm

how the probability of error goes to zero when the cross-over probability is 0.1229*.*

Is it possible to construct a reasonably good algorithm other than the sum-product algorithm so that the check node output always satisfies the symmetry condition? The answer is no. In fact, the sum-product algorithm is the only good algorithm that preserves the symmetry condition for check nodes with degree larger than two as we will show in the following lemma. For degree-2 check nodes, the sum-product algorithm is the same as the max-product algorithm. We first define a parity-preserving mapping.

Definition 5.1 *A mapping* $f : \mathbb{R}^n \to \mathbb{R}$ *is parity-preserving if*

$$f(x_1, x_2, \ldots, x_n) = \varphi(|x_1|, |x_2|, \ldots, |x_n|) \prod_{i=1}^{n} \operatorname{sgn}(x_i)$$

for some symmetric function $\varphi : \mathbb{R}^{+n} \to \mathbb{R}^+$*.*

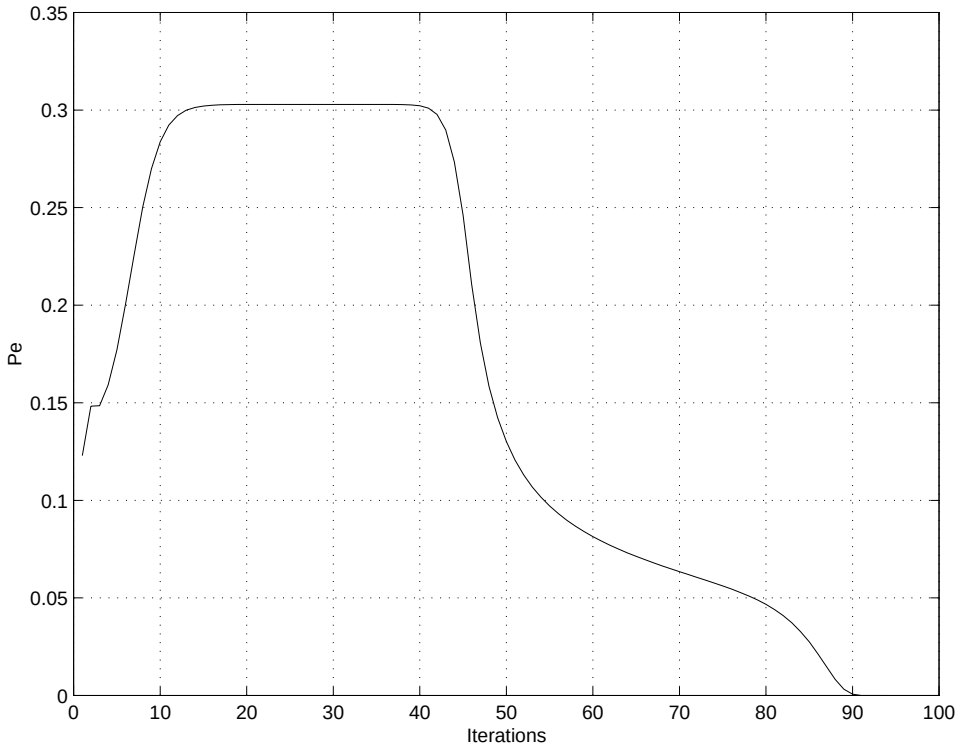


Figure 5-7: The probability of error for a (3,4)-regular LDPC code for the BSC using the max-product algorithm

A parity-preserving mapping produces correct estimates of bits when a hard decision is used.

Lemma 5.6 *The sum-product algorithm is the only parity-preserving mapping for a check node with degree larger than two that preserves the symmetry condition for all symmetric channels.*

Proof: Let d_c denote the degree of a check node. If input distributions are g_{p_i} for $1 \leq i \leq d_c$ and $0 < p_i < \frac{1}{2}$, then the d_c-th output of a parity-preserving mapping is given by the following:

$$(1-p)\delta\left(x-d\right)+p\delta\left(x+d\right), \tag{5.13}$$

for some $d \geq 0$, where p is given by the following:

$$\begin{aligned} p &= \frac{1}{2}\prod_{i=1}^{d_c-1}(1-p_i+p_i)-\frac{1}{2}\prod_{i=1}^{d_c-1}(1-p_i-p_i) \\ &= \frac{1}{2}-\frac{1}{2}\prod_{i=1}^{d_c-1}(1-2p_i). \end{aligned}$$

For the distribution (5.13) to satisfy the symmetry condition, d should satisfy the following:

$$d=\log\frac{1-p}{p},$$

which is equivalent to

$$d=2\tanh^{-1}\left(\prod_{i=1}^{d_c-1}\tanh\left(\frac{d_i}{2}\right)\right),$$

where $d_i=\log\frac{1-p_i}{p_i}$ for $1 \leq i \leq d_c-1$. Therefore, the mapping can only be the sum-product algorithm. Since the sum-product algorithm does preserve the symmetry condition for all symmetric channels the result follows. □

5.3.5 Thresholds and Optimization

Despite some shortcomings, there are some applications where the max-product algorithm is preferable. The max-product algorithm does not require the estimation of the channel noise (if it is converted to the min-sum algorithm and the costs are squared distances) and is simpler to execute than the sum-product algorithm. The max-product algorithm can then be used as a simplified decoding algorithm that approximates the sum-product algorithm. We show two examples where the max-product algorithm is only about 0.6 dB worse than the sum-product algorithm both in thresholds and in simulations.

Figure 5-8 shows thresholds and simulation results for a (3,6) code for the AWGN channel when the (a) sum-product and (b) max-product algorithm is used. For simulations, a block length 10^6 was used. The code graph was constructed randomly, except that cycles of length 2 and 4 were avoided. The threshold values are (a) 0.8809 ($E_b/N_0 \sim 1.1$ dB) and (b) 0.8223 ($E_b/N_0 \sim 1.7$ dB). Simulation results show excellent agreement with these theoretical predictions.

Table 5.3 shows two rate-1/2 codes optimized using (a) sum-product and (b) max-product algorithms for AWGN channels. Threshold values are shown in terms of both noise standard variation (σ) and dB gap from the Shannon limit ($\mathrm{SNR}_{\mathrm{norm}}$). The max-product algorithm is only about 0.6 dB worse than the sum-product algorithm for both (3,6) and $d_l = 20$ codes. Note that the two performances are quite different, as expected from the (3,6) code case. However, it is interesting that the improvement for each case is quite similar, i.e., the $d_l = 20$ code optimized for the sum-product algorithm is about 0.8 dB better than the (3,6) code when they are evaluated using the sum-product algorithm, and the $d_l = 20$ code optimized for the max-product algorithm is also about 0.8 dB better than the (3,6) code when they are evaluated using the max-product algorithm. It is also interesting that a code needs to be optimized for the target decoding algorithm for a better performance.

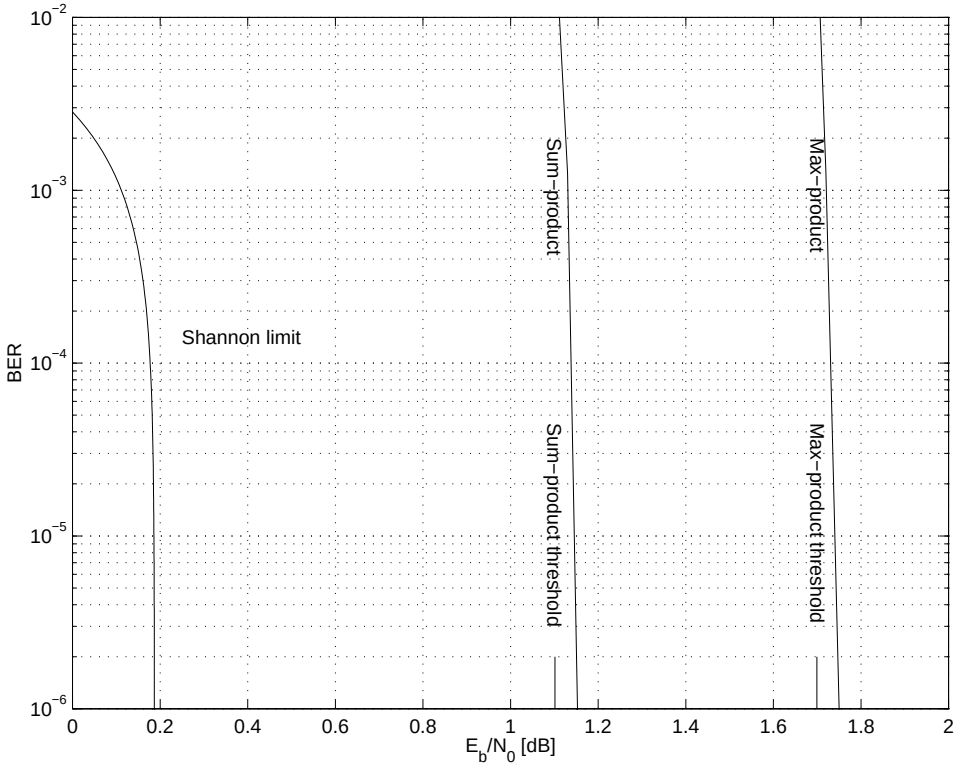


Figure 5-8: Simulation results for a (3,6) code for the AWGN channel using a block length of 10^6. The sum-product and max-product algorithms are used.

Table 5.3: Good rate-1/2 codes with $d_l = 20$ for the sum-product and max-product algorithms for the AWGN channel

Optimized for	sum-product	max-product
λ_2	0.234029	0.303702
λ_3	0.212425	0.277538
λ_6	0.146898	0.028432
λ_7	0.102840	0.200133
λ_{20}	0.303808	0.190195
ρ_{av}	8.28125	6.96875
Sum-product threshold σ	0.96693	0.92306
Sum-product SNR_{norm} [dB]	0.1051	0.5083
Max-product threshold σ	0.842	0.901
Max-product SNR_{norm} [dB]	1.31	0.71

5.4 Summary

In this chapter, we have introduced LDPC codes and the sum-product and max-product algorithms. We have shown the relationship between density evolution and iterative normalization. We have also shown how to develop discretized density evolution algorithm that will be used in Chapter 8 to design LDPC codes that are very close to the Shannon limit. Some interesting aspects of density evolution for the max-product algorithm have been shown.

Chapter 6

Capacity and Stability

In this chapter, we characterize symmetric channels using their capacity and stability functions. For the rest of this thesis, we consider only the density evolution for the sum-product algorithm.

We first give some examples of symmetric channels.

6.1 Channel Examples

We summarize relevant characteristics of some channels in the following examples.

Example 6.1 (Binary Erasure Channel (BEC)) *If the erasure probability is ϵ, then the channel capacity is given by*

$$C_{BEC}(\epsilon) = 1 - \epsilon.$$

The stability function of the BEC [61, 53] is given by

$$S_{BEC}(\epsilon) = \epsilon.$$

The initial LLR message density is given by

$$\epsilon\Delta_0 + (1-\epsilon)\Delta_\infty.$$

Example 6.2 (Binary Symmetric Channel (BSC)) *If p is the cross-over probability, then the channel capacity is given by*

$$C_{BSC}(p) = 1 - H(p),$$

where $H(\cdot)$ is the binary entropy function. The stability function of the BSC [53] is given by

$$S_{BSC}(p) = 2\sqrt{p(1-p)}.$$

The initial LLR message density is given by

$$p\Delta_{-\log\frac{1-p}{p}} + (1-p)\Delta_{\log\frac{1-p}{p}}.$$

The mean m_{BSC} and the variance σ^2_{BSC} of the initial message are given by

$$\begin{aligned} m_{BSC} &= (1-2p)\log\frac{1-p}{p} \\ \sigma^2_{BSC} &= 4p(1-p)\left(\log\frac{1-p}{p}\right)^2. \end{aligned}$$

We define the signal-to-noise ratio SNR_{BSC} of the initial message as

$$SNR_{BSC} = \frac{m^2_{BSC}}{\sigma^2_{BSC}} = \frac{4p^2-4p+1}{4p(1-p)}.$$

Example 6.3 (Laplace Channel) *If the input is ± 1 and the additive noise of the Laplace channel has a density*

$$p(z) = \frac{1}{2\lambda}e^{-|z|/\lambda},$$

then the channel capacity is given by

$$C_{Laplace}(\lambda) = \frac{4\arctan e^{-1/\lambda} - \pi}{2e^{1/\lambda}\ln 2} - \log_2\left(\frac{1+e^{-2/\lambda}}{2}\right),$$

The stability function of the Laplace channel [53] is

$$S_{Laplace}(\lambda) = \frac{1+\lambda}{\lambda}e^{-1/\lambda}.$$

The initial LLR message density is given by

$$\frac{1}{2}\Delta_{\frac{2}{\lambda}} + \frac{1}{2}e^{-2/\lambda}\Delta_{-\frac{2}{\lambda}} + \frac{1}{4}e^{u/2-1/\lambda}I_{|u|\leq\frac{2}{\lambda}},$$

where $I_{|u|\leq\frac{2}{\lambda}} = 1$ *if* $|u| \leq \frac{2}{\lambda}$ *and 0 otherwise. The mean* $m_{Laplace}$ *and the variance* $\sigma^2_{Laplace}$ *of the initial message are given by*

$$\begin{aligned} m_{Laplace} &= \frac{2}{\lambda} - 1 + e^{-2/\lambda} \\ \sigma^2_{Laplace} &= 3 - e^{-4/\lambda} - 2e^{-2/\lambda} - \frac{8}{\lambda}e^{-2/\lambda}. \end{aligned}$$

The signal-to-noise ratio $SNR_{Laplace}$ *of the initial message is*

$$SNR_{Laplace} = \frac{m^2_{Laplace}}{\sigma^2_{Laplace}}.$$

Example 6.4 (AWGN Channel) *The channel capacity is given by*

$$C_{AWGN}(\sigma_n^2) = -\int_{\mathbb{R}} \tilde{f}(x)\log_2\tilde{f}(x)dx - \frac{1}{2}\log_2 2\pi e\sigma_n^2,$$

where the marginal output distribution $\tilde{f}(x)$ *is given by*

$$\tilde{f}(x) = \frac{\mathcal{N}(x;1,\sigma_n^2) + \mathcal{N}(x;-1,\sigma_n^2)}{2},$$

where $\mathcal{N}(x; m, \sigma_n^2)$ *is a Gaussian density with mean* m *and variance* σ_n^2*. The stability function of the AWGN channel [53] is*

$$S_{AWGN}(\sigma_n^2) = \exp\left(-\frac{1}{2\sigma_n^2}\right).$$

The initial LLR message density is given by

$$\mathcal{N}(u; \frac{2}{\sigma_n^2}, \frac{4}{\sigma_n^2}) = \frac{\sigma_n}{\sqrt{8\pi}} e^{-\frac{(u-2/\sigma_n^2)^2\sigma_n^2}{8}},$$

where $m_{AWGN} = \frac{2}{\sigma_n^2}$ *is the mean and* $\sigma_{AWGN}^2 = \frac{4}{\sigma_n^2}$ *is the variance of the initial message. The signal-to-noise ratio* SNR_{AWGN} *of the initial message is*

$$SNR_{AWGN} = \frac{m_{AWGN}^2}{\sigma_{AWGN}^2} = \frac{1}{\sigma_n^2}.$$

Our analysis should hold for other types of binary-input memoryless symmetric-output channels whose capacity decreases monotonically as the channel parameter increases (to exclude degenerate and ill-behaved cases). The inverse of a capacity function exists for all rates from 0 to 1 for these channels, which includes all of our channel examples.

6.2 Maximum-Stability Functions

In this section, we introduce maximum-stability functions, useful tools in comparing different symmetric channels.

Definition 6.1 *For a channel with the capacity function* $C(x)$ *and the stability function* $S(x)$*, we define the maximum-stability function* $\xi(r)$ *as*

$$\xi(r) = S(C^{-1}(r)),$$

where $0 \leq r \leq 1$*.*

Note that the maximum-stability function is a property of a channel only, i.e., it does not depend on any code.

The following lemma shows how the maximum-stability function is related to the bound on the performance of a code.

Lemma 6.1 *If* $\lambda'(0)\rho'(1) > \left[\xi_X\left(1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx}\right)\right]^{-1}$, *then there exists* $\exists\varepsilon > 0$ *such that no* $(\lambda(x), \rho(x))$ *code can achieve the channel capacity of the channel* X *within* ε, *where* $\xi_X(\cdot)$ *is the maximum-stability function of the channel.*

In Figure 6-1, we show the maximum-stability functions for various channels including component channels at the top (noisiest) level of several multilevel channels, namely, mod-AWGN and 4-PAM, in Section 8.2. We observe that the BEC has the lowest $\xi(r)$ compared to the other curves, especially at low rates, where $\xi(r)$ for the BEC is very different from the others. This seems to suggest that the BEC is the easiest channel of the four channels for code optimization purposes. This is especially true when codes are marginally stable, which is the case for most good codes we have constructed so far. Surprisingly the AWGN channel, which is the worst additive white noise channel of given variance, is the next easiest channel in this sense when the rate is greater than 0.6.

Figure 6-1 seems to suggest the BEC has the lowest and the BSC has the highest maximum-stability functions among all symmetric channels. This is actually true as we prove in Theorem 6.4. We first prove the following two lemmas.

Lemma 6.2 $H(p) \le 2\sqrt{p(1-p)}$ *for all* $0 \le p \le \frac{1}{2}$, *with equality iff* $p = 0$ *or* $\frac{1}{2}$, *where* $H(p)$ *is the binary entropy function.*

Proof: Let $t(p) = 2\sqrt{p(1-p)} - H(p)$, then $t(p) = 0$ if $p = 0$ or $\frac{1}{2}$. Furthermore, $t'(0) = \infty$ and $t'(1/2) = 0$. Since $t''(p) = \frac{2\sqrt{p(1-p)}\log_2 e - 1}{2\{p(1-p)\}^{3/2}}$ has only one zero in $(0, 1/2)$ and $t''(1/2) > 0$, the result follows. □

Lemma 6.3

$$r(p,q) = w'(q)\{H(p) - H(q)\} - H'(q)\{w(p) - w(q)\} \ge 0, \quad \textit{for all } 0 \le p, q \le \tfrac{1}{2},$$

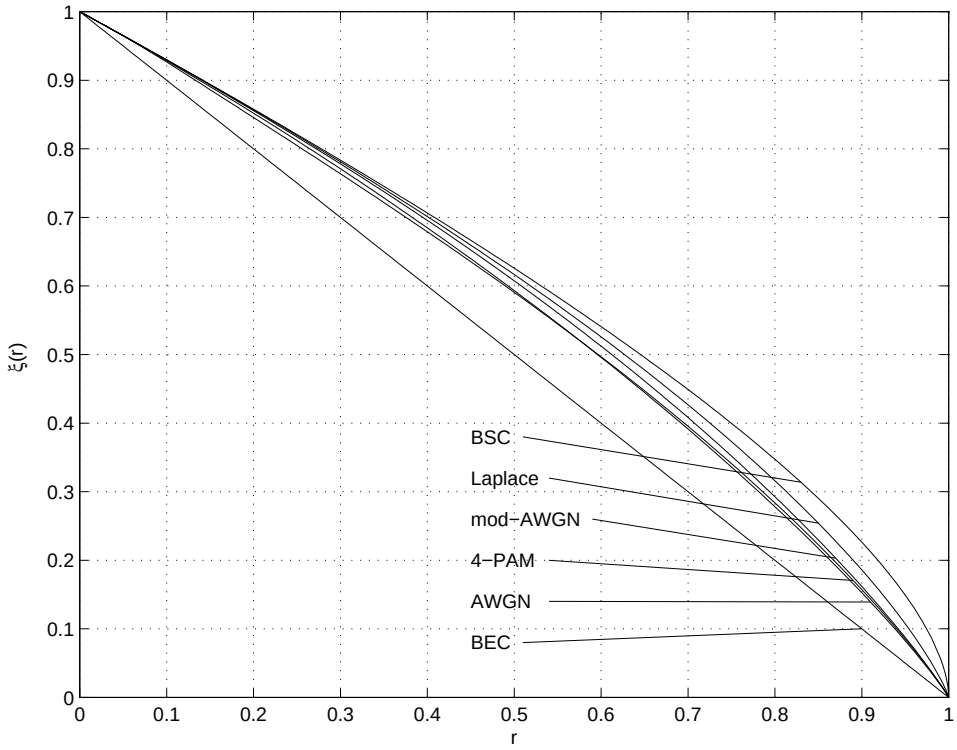


Figure 6-1: Maximum-stability functions $\xi(r)$ for various channels as functions of the rate r

where $w(p) = 2\sqrt{p(1-p)}$.

Proof: First observe that $r(p,p) = 0$ for all $0 \leq p \leq \frac{1}{2}$. $\frac{\partial r(p,q)}{\partial p} = w'(q)H'(p) - w'(p)H'(q)$ is 0 if $p = q, \frac{1}{2}$ and is $-\infty$ if $p = 0$. Since

$$\frac{\partial^2 r(p,q)}{\partial p^2} = \frac{1}{p(1-p)} \left(\frac{H'(q)}{2\sqrt{p(1-p)}} - w'(q) \log_2 e \right)$$

has at most one zero in $0 < p < \frac{1}{2}$ when $0 < q \leq \frac{1}{2}$, the result follows. $\square$

Theorem 6.4 *The BEC has the lowest and the BSC has the highest maximum-stability functions among all symmetric channels for all rates from zero to one.*

Proof: We use the channel capacity C and stability function S as functions of a compact initial density $f(\cdot)$ as given in (4.11) and (5.6), i.e., C and S are given by

the following:

$$\begin{aligned} C &= 1 - \int_{\left[0,\frac{1}{2}\right]^+} f(p)H(p)\,dp \\ S &= \int_{\left[0,\frac{1}{2}\right]^+} f(p)2\sqrt{p(1-p)}\,dp. \end{aligned}$$

Finding the minimum or maximum of the maximum-stability function $\xi(r)$ for each rate r over all $f(p)$ that satisfies (4.10) is equivalent to the following linear programming:

$$\begin{aligned} \min_{f(p)} \quad & \pm \int_{\left[0,\frac{1}{2}\right]^+} f(p)2\sqrt{p(1-p)}\,dp \\ \text{subject to} \quad & \int_{\left[0,\frac{1}{2}\right]^+} f(p)\,dp = 1, \\ & \int_{\left[0,\frac{1}{2}\right]^+} f(p)H(p)\,dp = 1 - r, \\ & f(p) \geq 0, \end{aligned}$$

where $\pm$ corresponds to minimization $(+)$ and maximization $(-)$ problems, respectively.

This is an infinite-dimensional linear programming. For finite-dimensional linear programming, we can always use the weak duality theorem [5] to easily find solutions to the primal and dual problems since there is no gap between the two spaces of the objective functions. In general, this is not true when the dimension is infinite [2]. However, as we will see later, there is no "duality gap" in our problem and feasible solutions to the primal and dual problems of equal cost are indeed optimal solutions to both problems.

The dual problem of the above problem is given by the following:

$$\begin{aligned} \max_{y_1,y_2} \quad & y_1 + y_2(1-r) \\ \text{subject to} \quad & y_1 + y_2 H(p) \leq \pm 2\sqrt{p(1-p)}, \text{ for } 0 \leq p \leq \tfrac{1}{2}. \end{aligned}$$

The following are feasible solutions to the primal and dual problems of equal cost for minimization:

$$\begin{aligned} f(p) &= r\delta(p) + (1-r)\delta\left(p - \frac{1}{2}\right), \\ (y_1, y_2) &= (0, 1), \end{aligned}$$

which can be easily verified using Lemma 6.2. Therefore, since there is no duality gap, this is the optimal solution to both problems. Note that this $f(p)$ is the compact initial density of the BEC with erasure probability $1-r$. Thus, we conclude that the BEC minimizes the maximum-stability function among all symmetric channels for all rates.

For maximization, the following are feasible solutions with equal cost to the primal and dual problems:

$$\begin{aligned} f(p) &= \delta\left(p - q\right) \\ y_2 &= -\frac{1-2q}{\sqrt{q(1-q)}\log_2\frac{1-q}{q}} \\ y_1 &= -y_2 H(q) - 2\sqrt{q(1-q)}, \end{aligned}$$

where $q = H^{-1}(1-r)$. It is easy to check that two costs are the same and the above $f(p)$ is a feasible solution to the primal problem. The constraint of the dual problem becomes

$$\frac{w'(q)}{H'(q)}\left(H(p) - H(q)\right) \geq w(p) - w(q),$$

where $w(p) = 2\sqrt{p(1-p)}$ for $0 \leq p \leq \frac{1}{2}$. It follows from Lemma 6.3 that this holds for all $0 < p, q < \frac{1}{2}$. Thus, these are optimal solutions to both primal and dual problems since there is no duality gap. Note that this $f(p)$ is the compact initial density of the BSC with a cross-over probability q. Thus, we conclude that the BSC maximizes the maximum-stability function among all symmetric channels for all rates. $\square$

For the BEC, $\xi'(0) = -1$. It seems from Figure 6-1 that $\xi'(0)$ is the same for the other channels. In fact, this is true, as is stated in the following theorem.

Theorem 6.5 *If the initial density of a symmetric channel does not have a point mass at infinity, then* $\xi'(0) = -\ln 2$.

Proof: The compact initial density $f(p)$ does not have any delta function at zero for this case. The channel capacity is given by

$$C = \int_0^{\frac{1}{2}} f(p)\,[1 - H(p)]\ dp.$$

Let $C = \varepsilon \ll 1$. Let $x = H^{-1}\,(1 - \sqrt{\varepsilon})$. Then, we have the following inequalities:

$$\begin{aligned} \varepsilon &\geq \int_0^x f(p)\,[1 - H(p)]\ dp \\ &\geq [1 - H(x)] \int_0^x f(p)\ dp \\ &= \sqrt{\varepsilon} \int_0^x f(p)\ dp. \end{aligned}$$

Therefore, the integral of $f(p)$ from 0 to x is less than or equal to $\sqrt{\varepsilon}$, i.e.,

$$\begin{aligned} \sqrt{\varepsilon} &\geq \int_0^x f(p)\ dp \\ &\geq \int_0^x f(p)\,[1 - H(p)]\ dp, \end{aligned}$$

where the second inequality follows from $1 - H(p) \leq 1$.

Since the delta function at zero is the only initial density for zero capacity, we get $\xi(0) = 1$ for any symmetric channel. Therefore, the change in the stability function when the rate is increased from zero is given by the following:

$$1 - S = \int_0^{\frac{1}{2}} f(p) \left[1 - 2\sqrt{p(1-p)}\right]\ dp.$$

Since $1 - H(p) \leq 2\left(1 - 2\sqrt{p(1-p)}\right)$ for all $0 \leq p \leq \frac{1}{2}$, we conclude that $1 - S \geq \frac{\varepsilon}{2}$.

Similarly, using $1 - 2\sqrt{p(1-p)} \leq 1$ for $0 \leq p \leq \frac{1}{2}$, we get

$$\sqrt{\varepsilon} \geq \int_0^x f(p)\, dp \geq \int_0^x f(p) \left[1 - 2\sqrt{p(1-p)}\right]\, dp.$$

As $\varepsilon \to 0$, we see that we need only the contributions from x to $\frac{1}{2}$ for both integrals for C and $1 - S$. Since $x \to \frac{1}{2}$ as $\varepsilon \to 0$, we get

$$\begin{aligned}
-\xi'(0) &= \lim_{\varepsilon \to 0} \frac{1-S}{C} \\
&= \lim_{x \to \frac{1}{2}} \frac{1 - 2\sqrt{x(1-x)}}{1 - H(x)} \\
&= \lim_{x \to \frac{1}{2}} \frac{\frac{\partial^2 \{1 - 2\sqrt{x(1-x)}\}}{\partial x^2}}{\frac{\partial^2 \{1 - H(x)\}}{\partial x^2}} \\
&= \lim_{x \to \frac{1}{2}} \ln 2 \frac{1}{2\sqrt{x(1-x)}} \\
&= \ln 2.
\end{aligned}$$

□

We note that the above proof should hold for initial densities with a point mass at infinity if the mass becomes sufficiently small compared to the other contributions to the channel capacity.

Corollary 6.6 *For the BSC, mod-AWGN, the top (noisiest) level of 4-PAM, Laplace, and AWGN channels,* $\xi'(0) = -\ln 2$.

Proof: The initial densities of these channels do not have a point mass at infinity. □

Corollary 6.7 $-1 \leq \xi'(0) \leq -\ln 2$ *for any symmetric channel.*

Proof: $\xi'(0) = -1$ for the BEC and $\xi'(0) = -\ln 2$ for the BSC. □

6.3 Mapping of Thresholds

In this section, we investigate some interesting characteristics of thresholds of LDPC codes on various binary-input symmetric-output memoryless channels including the

Table 6.1: Channel capacity at the threshold values of various (j,k)-regular LDPC codes. For each code, threshold values in terms of the channel parameters, ϵ, p, λ, and σ_n, are evaluated using density evolution for the BEC, the BSC, the Laplace and AWGN channels, respectively. These threshold values are used to calculate the channel capacity of the corresponding channels.

j	k	rate	$C_{\text{BEC}}(\epsilon)$	$C_{\text{BSC}}(p)$	$C_{\text{Laplace}}(\lambda)$	$C_{\text{AWGN}}(\sigma_n^2)$
3	6	0.5	0.571	0.584	0.577	0.571
4	8	0.5	0.617	0.609	0.606	0.605
5	10	0.5	0.658	0.641	0.640	0.643
3	5	0.4	0.482	0.489	0.485	0.480
4	6	1/3	0.494	0.479	0.478	0.479
3	4	0.25	0.353	0.349	0.348	0.346
4	10	0.6	0.692	0.689	0.684	0.683
3	9	2/3	0.717	0.739	0.728	0.720
3	12	0.75	0.790	0.814	0.801	0.793

BEC, the BSC, and the Laplace and AWGN channels. These characteristics will simplify the analysis of sum-product decoding of LDPC codes and useful principles for designing good LDPC codes.

Table 6.1 shows the channel capacity of each channel evaluated at the thresholds of various (j,k)-regular codes. For each code, the threshold values in terms of the channel parameter, ϵ, p, λ, and σ_n, are evaluated using density evolution for the BEC, the BSC, and the Laplace and AWGN channels, respectively. These threshold values are used to calculate the channel capacity of the corresponding channels.

The first thing we notice is that the channel capacity values are very similar across the channels. As we will show later, the same is true for various irregular codes. This is surprising since the thresholds for each channel are calculated by running density evolution, and since each channel has a different initial message density, one would expect that density evolution would behave very differently and that there would be no strong correlation across various channels.

The rate of each code in Table 6.1 is smaller than the corresponding capacity values in the columns on the right. This difference can be interpreted as a rate penalty for imposing an LDPC structure and message-passing decoding on an optimal random

code. What Table 6.1 shows is that this penalty is similar across the channels for these cases.

One immediate application of this observation is that we can predict threshold values reasonably well for other channels, given just the threshold value for one channel. It seems that density evolution is not sensitive to the initial message density, but rather sees the channel capacity as the most important parameter of the channel. This may not seem obvious at first; one would think that the mean, the variance, or the SNR of the initial message density would be more important. As we will show later, this is not true; the "channel capacity" is a more important parameter. This observation has the following implications and applications.

1. This observation provides a good explanation of why codes optimized for one channel are usually good for other channels, as observed in [53].

2. The calculation of thresholds for any binary-input memoryless symmetric channel can be carried out approximately (but reasonably accurately) by using the simplest channel, namely the BEC. We call this the "erasure-channel approximation." The resulting threshold value needs to be mapped to the correct domain by using capacity functions and their inverses.

3. Designing good degree distributions can also be done efficiently using the simplest channel, namely the BEC, provided that the BEC stability condition is replaced by that of the target channel, since the stability condition of the BEC is very different from those of the other three channels, as we will show later.

4. We can understand density evolution and therefore the sum-product algorithm reasonably well using the simplest channel, namely the BEC. (Again, the stability condition should be replaced.)

5. This observation is the key motivation for designing an improved approximation method for modeling density evolution for the AWGN channel, which we call the "reciprocal-channel approximation." This is about 10 times more accurate than the erasure-channel approximation.

To demonstrate the usefulness of this observation and also to show that the channel capacity provides accurate mappings between threshold values for various channels, we show threshold values of various codes and channels in Figures 6-2–6-6. Various regular codes in Table 6.1 are indicated by ($\circ$), and irregular codes designed for AWGN channels with rates 0.1 to 0.9 are indicated by $d_l = 10$ ($\square$) and $d_l = 20$ ($\triangle$), respectively. Also shown in the figures are equal-variance, equal-mean, equal-stability, equal-capacity, and equal-SNR curves, where variances, means, and SNRs are for initial messages as defined in the previous channel examples (these are not available for the BEC). For example, equal-capacity curves are obtained by setting two channel capacities as equal; e.g., in Figure 6-2, the equal-capacity curve is defined by the following mapping:

$$\sigma_n^2 = C_{\text{AWGN}}^{-1}\left(C_{\text{BSC}}(p)\right).$$

Other curves are obtained similarly.

Some observations are made here:

1. The equal-capacity curves are monotonically increasing, which is due to the monotonicity of the channel capacity functions.

2. The equal-capacity curves are the most accurate for mapping thresholds from one channel to another.

3. The equal-stability curves are also accurate for such mappings, except for cases involving the BEC.

4. The equal-mean, equal-variance, and equal-SNR curves are not good for mapping thresholds from one channel to another.

5. The accuracy of the threshold mapping using equal-capacity curves is very good for both irregular and regular codes (0.06 to 5 dB from the AWGN capacity) and for all rates (0.1 to 0.9).

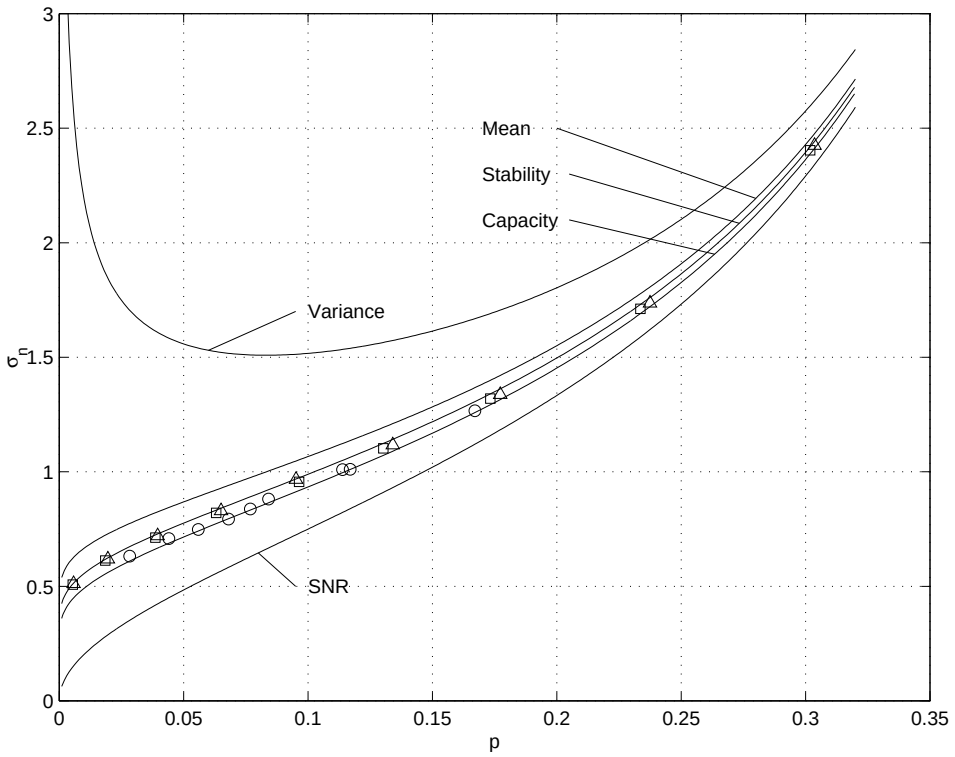


Figure 6-2: BSC and AWGN thresholds for various regular ($\bigcirc$) and irregular codes with rates 0.1 to 0.9 designed for AWGN channels with $d_l = 10$ ($\square$) and $d_l = 20$ ($\triangle$). The curves are (from top to bottom) equal-variance, equal-mean, equal-stability, equal-capacity, and equal-SNR. The variances, means, and SNRs are for initial messages.

6. Codes optimized for one channel (the AWGN channel in this case) are also good for other channels. This is because if all codes lie close to the equal-capacity curves, then optimizing a code for one channel results in higher thresholds for other channels, too. When the code is near the channel capacity of one channel, it will be also near the channel capacity of other channels.

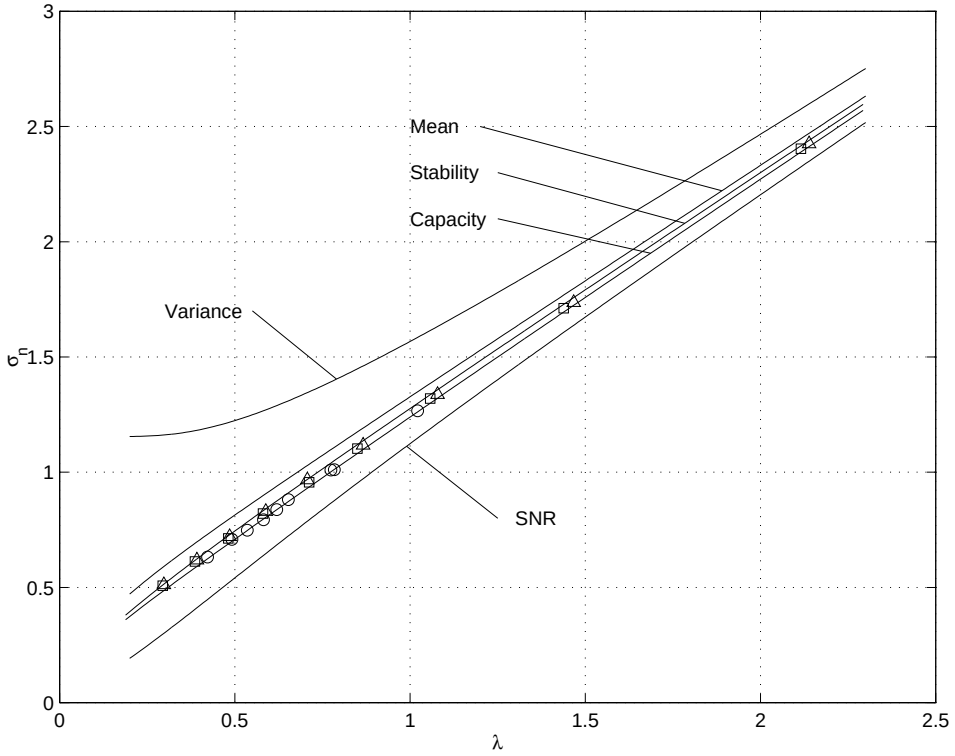


Figure 6-3: Laplace and AWGN thresholds for various regular ($\bigcirc$) and irregular codes with rates 0.1 to 0.9 designed for AWGN channels with $d_l = 10$ ($\square$) and $d_l = 20$ ($\triangle$). The curves are (from top to bottom) equal-variance, equal-mean, equal-stability, equal-capacity, and equal-SNR. The variances, means, and SNRs are for initial messages.

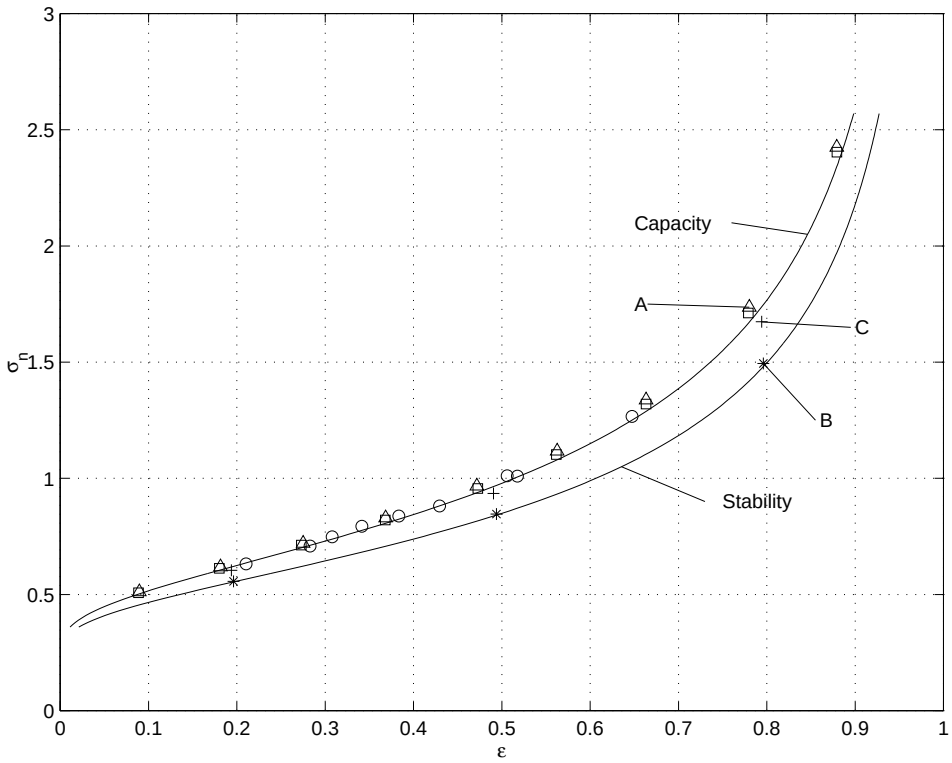


Figure 6-4: BEC and AWGN thresholds for various regular ($\circ$) and irregular codes with rates 0.1 to 0.9 designed for AWGN channels with $d_l = 10$ ($\square$) and $d_l = 20$ ($\triangle$). The curves are (from top to bottom) equal-capacity and equal-stability. ($*$) denotes irregular codes designed for the BEC. ($+$) denotes irregular codes designed for the BEC using the AWGN stability condition. A($\triangle$) is a rate-0.2 irregular code designed for the AWGN channel, B($*$) is a rate-0.2 irregular code designed for the BEC, and C($+$) is a rate-0.2 irregular code designed for the BEC using the AWGN stability condition. All ($*$,$+$) codes have $d_l = 20$.

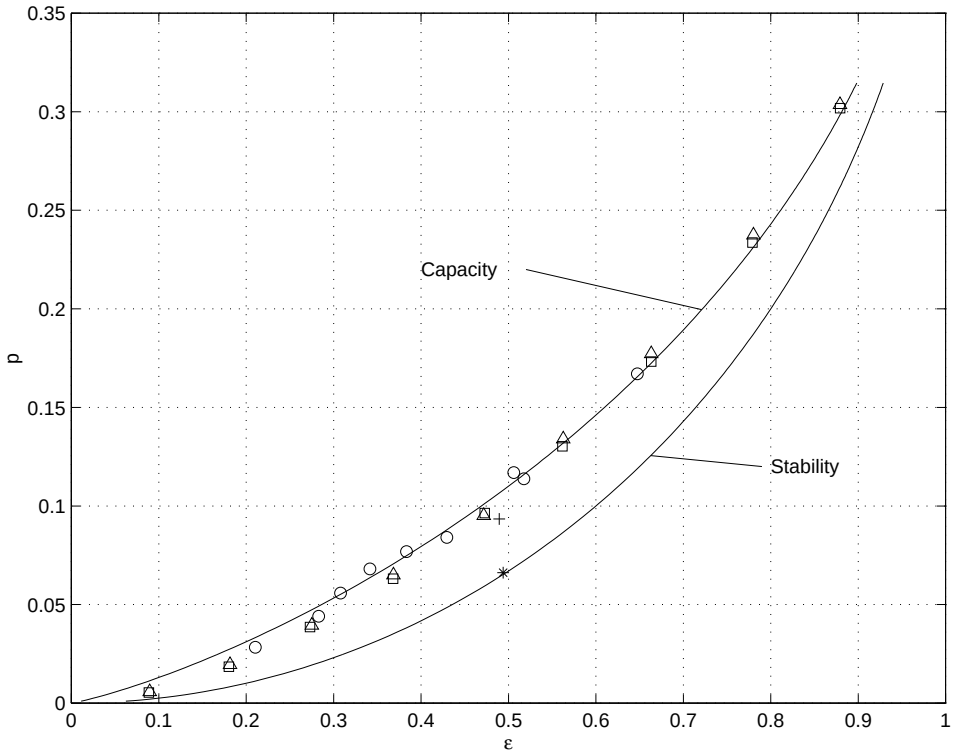


Figure 6-5: BEC and BSC thresholds for various regular ($\circ$) and irregular codes with rates 0.1 to 0.9 designed for AWGN channels with $d_l = 10$ ($\square$) and $d_l = 20$ ($\triangle$). The curves are (from top to bottom) equal-capacity and equal-stability. ($*$) denotes an irregular code designed for the BEC. ($+$) denotes an irregular code designed for the BEC using the BSC stability condition. All ($*$,$+$) codes have $d_l = 20$.

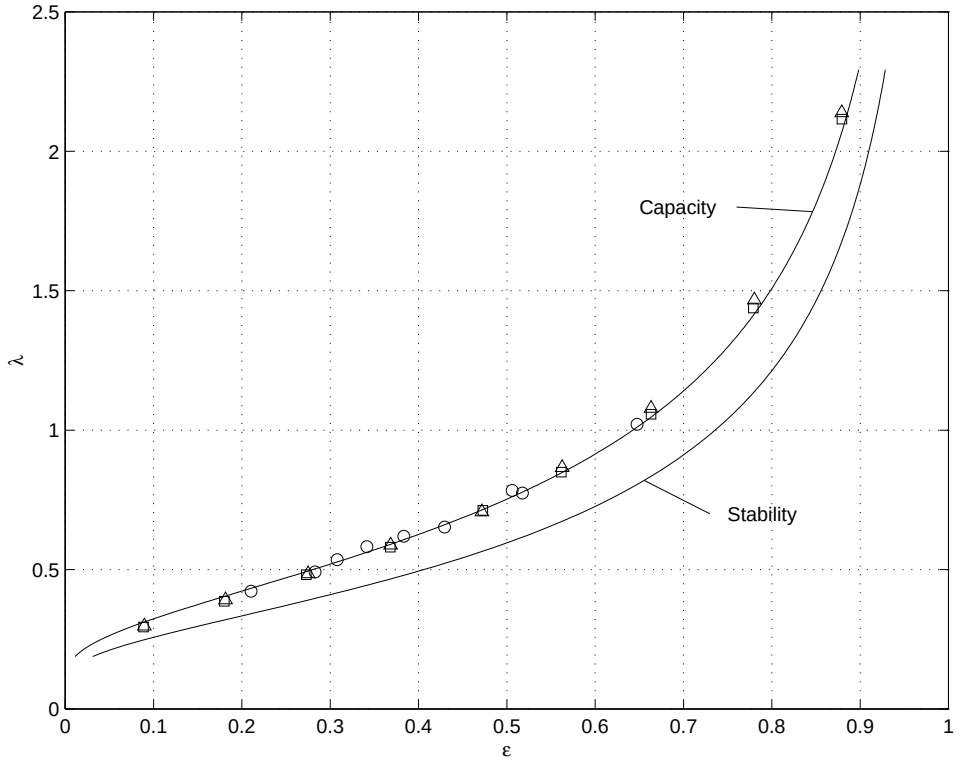


Figure 6-6: BEC and Laplace thresholds for various regular ($\bigcirc$) and irregular codes with rates 0.1 to 0.9 designed for AWGN channels with $d_l = 10$ ($\square$) and $d_l = 20$ ($\triangle$). The curves are (from top to bottom) equal-capacity and equal-stability.

6.4 Summary

In this chapter, we have developed some tools to compare various symmetric channels. We have defined the maximum-stability function for symmetric channels and have shown some of its properties. We have shown some empirical observations on threshold values for different channels and discussed the implications. These observations will be used as motivations for developing several approximation methods in Chapter 7.

Chapter 7

Approximation Methods

7.1 Erasure-Channel Approximation and Optimization

Binary erasure channels are very simple. Density evolution is simply described using an erasure probability, which becomes a one-dimensional evolution [45, 44]. Due to this simplicity, it is even possible to construct a capacity-achieving sequence of degree distributions for BECs [44].

Using the observations made in the previous section, we can use the BEC to analyze density evolution for some other symmetric channels. We show that we can calculate thresholds accurately using this method, which we call the "erasure-channel approximation." We also show that this approximation can be used to optimize degree distributions for other channels as well, provided that we modify the BEC stability condition.

We first show how threshold values can be calculated accurately using the erasure-channel approximation, where the erasure threshold ϵ^* of a (j,k)-code is calculated, and the threshold value is mapped to the threshold value of the original channel using the equal-capacity curve for the two channels, i.e.,

$$x = C_{\mathrm{X}}^{-1}(C_{\mathrm{BEC}}(\epsilon^*)),$$

Table 7.1: Exact and approximate (erasure-channel approximation σ_{ECA}, Gaussian approximation σ_{GA}, and reciprocal-channel approximation σ_{RCA}) threshold values for various (j,k)-regular LDPC codes for the binary-input AWGN channel and sum-product decoding. All threshold values are rounded down.

j	k	rate	σ_{exact}	σ_{ECA}	error[dB]	σ_{GA}	error[dB]	σ_{RCA}	error[dB]
3	6	0.5	0.8809	0.8811	0.001	0.8747	0.06	0.8808	0.001
4	8	0.5	0.8376	0.8242	0.14	0.8323	0.06	0.8383	0.007
5	10	0.5	0.7936	0.7757	0.20	0.7910	0.03	0.7948	0.014
3	5	0.4	1.0093	1.0054	0.03	1.0003	0.08	1.0091	0.001
4	6	1/3	1.0109	0.9879	0.20	1.0035	0.06	1.0116	0.006
3	4	0.25	1.2667	1.2502	0.11	1.2517	0.10	1.2658	0.006
4	10	0.6	0.7481	0.7386	0.11	0.7440	0.05	0.7485	0.005
3	9	2/3	0.7082	0.7116	0.04	0.7051	0.04	0.7079	0.003
3	12	0.75	0.6320	0.6359	0.05	0.6297	0.03	0.6316	0.006

where x is the estimated threshold for the channel X and $C_{\mathrm{X}}(\cdot)$ is the capacity function of X. Tables 7.1, 7.2, and 7.3 show the accuracy of this approximation for various regular codes for AWGN, Laplace channels, and the BSC.

The erasure-channel approximation works very well in predicting thresholds for the AWGN and Laplace channels. To use the erasure-channel approximation for optimizing degree distributions, however, we need to modify the stability condition of the BEC, because the BEC is different from other channels in that the stability condition is very loose.

Figure 6-4 shows some examples of codes optimized for both AWGN channels and BECs. It shows BEC and AWGN thresholds for various regular ($\circ$) and irregular codes with rates 0.1 to 0.9 designed for AWGN channels with $d_l = 10$ ($\square$) and $d_l = 20$ ($\triangle$). ($*$) denotes irregular codes designed for the BEC. ($+$) denotes irregular codes designed for the BEC using the AWGN stability condition.

It is interesting to observe that when we optimize a code for the BEC, it is not near the equal-capacity line as shown in the figure as ($*$). For example, A($\triangle$) is a code designed for the AWGN channel and B($*$) is a code designed for the BEC, where both codes have rate $= 0.2$ and $d_l = 20$. We note that A is closer to the AWGN capacity than B and B is closer to the BEC capacity than A.

Table 7.2: Exact and approximate (erasure-channel approximation λ_{ECA}, Gaussian approximation λ_{GA}, and reciprocal-channel approximation λ_{RCA}) threshold values for various (j,k)-regular LDPC codes for the binary-input Laplace channel and sum-product decoding. All threshold values are rounded down.

j	k	rate	λ_{exact}	λ_{ECA}	error(%)	λ_{GA}	error(%)	λ_{RCA}	error(%)
3	6	0.5	0.6525	0.660	1.2	0.654	0.3	0.660	1.2
4	8	0.5	0.6194	0.607	2.0	0.614	0.7	0.620	0.2
5	10	0.5	0.5818	0.562	3.4	0.576	1.0	0.579	0.3
3	5	0.4	0.7741	0.777	0.5	0.772	0.2	0.781	0.9
4	6	1/3	0.7839	0.761	2.9	0.775	1.0	0.783	0.1
3	4	0.25	1.0212	1.010	1.0	1.011	0.9	1.025	0.4
4	10	0.6	0.5355	0.527	1.5	0.532	0.6	0.536	0.2
3	9	2/3	0.4917	0.502	2.2	0.496	1.0	0.499	1.5
3	12	0.75	0.4221	0.432	2.5	0.426	1.1	0.428	1.5

Table 7.3: Exact and approximate (erasure-channel approximation p_{ECA}, Gaussian approximation p_{GA}, and reciprocal-channel approximation p_{RCA}) threshold values for various (j,k)-regular LDPC codes for the BSC and sum-product decoding. All threshold values are rounded down.

j	k	rate	p_{exact}	p_{ECA}	error(%)	p_{GA}	error(%)	p_{RCA}	error(%)
3	6	0.5	0.0840	0.0879	4.5	0.0864	2.8	0.0878	4.4
4	8	0.5	0.0768	0.0747	2.7	0.0766	0.2	0.0780	1.6
5	10	0.5	0.0680	0.0635	6.5	0.0671	1.4	0.0679	0.1
3	5	0.4	0.1138	0.1159	1.9	0.1148	0.9	0.1167	2.6
4	6	1/3	0.1169	0.1120	4.2	0.1155	1.2	0.1173	0.3
3	4	0.25	0.1670	0.1655	0.9	0.1658	0.7	0.1684	0.8
4	10	0.6	0.0558	0.0551	1.2	0.0563	0.9	0.0573	2.8
3	9	2/3	0.0440	0.0491	12	0.0477	8.4	0.0483	9.8
3	12	0.75	0.0283	0.0332	17	0.0320	13	0.0323	14

We also observe that B is marginally stable for the BEC, i.e., its BEC threshold is 0.796 and it becomes unstable if $\epsilon > 0.799$, which means the BEC optimizer nearly fully utilizes the BEC stability condition. Since the BEC stability condition is loose, as indicated in Figure 6-1, when we use code B for the AWGN channel, the AWGN stability condition becomes dominant in determining its AWGN threshold. This means that B cannot be too far from the equal-stability curve in the direction of increasing σ_n, which is the direction of decreasing stability for the AWGN channel. This is because B becomes unstable for the BEC if $\epsilon > 0.799$ and therefore B becomes unstable for the AWGN channel if $\sigma_n > 1.493$ according to the equal-stability curve. In fact, it is very close to the equal-stability curve.

Now, it seems that using a BEC optimizer naively for other channels does not work well because the BEC stability condition is loose. This motivates us to replace the BEC stability condition with the stability condition of the target channel for optimizing codes using the BEC optimizer. In fact, this works very well and a code optimized this way is shown in Figure 6-4 as C(+), where the same parameters were used, i.e., rate = 0.2 and $d_l = 20$. Since designing a code for the BEC is much simpler than designing a code for the AWGN channel, this gives us an efficient way of optimizing codes for complex channels. What is even more interesting is that C is reasonably good for both AWGN and BEC channels, which was not possible using the optimization algorithms that considered only one channel.

7.2 Gaussian Approximation

In this section, we present a simple method for estimating the threshold for irregular LDPC codes on memoryless binary-input continuous-output AWGN channels with sum-product decoding. This method is based on approximating message densities as Gaussians (for regular LDPC codes) or Gaussian mixtures (for irregular LDPC codes) [12]. We show that, without much sacrifice in accuracy, a one-dimensional quantity, namely the mean of a Gaussian density, can act as a faithful surrogate for the message density, which is an infinite-dimensional vector.

Since this method is easier to analyze and computationally faster than density evolution, it can be a useful tool for understanding the behavior of the decoder and for optimizing irregular codes. For example, we show how to determine the rate of convergence of the error probability, and why there is an alternation between fast and slow decoding stages, as noted in [53]. We also use this method to find good irregular codes using linear programming. This algorithm not only optimizes degree distributions several orders of magnitude faster, but it is often almost as good as the optimization methods based on density evolution given in [53].

For turbo codes, a one-dimensional approximation for densities was first used by ten Brink [64]. However, he concentrated only on parallel concatenated codes and did not optimize codes. A similar idea was later used in [16] for turbo decoding including serial concatenated codes, which appeared at the same time as [12]. Since there is no simple formula for updating message densities for turbo decoding, Monte Carlo simulations were used to analyze approximate trajectories for Gaussian messages under turbo decoding in both papers [64, 16]. Similar to the mean as used in this paper, they used one-dimensional quantities, namely mutual information [64] and SNR [16], to approximate message densities. They showed that Gaussian approximation works reasonably well for turbo decoding, and argued that the position of the turbo waterfall region can be estimated by visualizing the trajectories.

In other related works, the sum-product algorithm was analyzed for graphs with cycles when messages are jointly Gaussian [23], [57], [71]. Since a Gaussian density is completely characterized by its mean vector and covariance matrix, the analysis of the sum-product algorithm becomes tractable. The main purpose of these works is to analyze how well decoding works on graphs with cycles. Surprisingly, the means converge to the correct posterior means when certain conditions are satisfied, even when there are cycles in the graph.

7.2.1 Gaussian Approximation for Regular LDPC Codes

The LLR message u_0 from the channel is Gaussian with mean $2/\sigma_n^2$ and variance $4/\sigma_n^2$, where σ_n^2 is the variance of the channel noise. Thus, if all $\{u_i, i \geq 1\}$ (which are iid)

are Gaussian in (5.1), then the resulting sum is also Gaussian because it is the sum of independent Gaussian random variables. Even if the inputs are not Gaussian, by the central limit theorem, the sum would look like a Gaussian if many independent random variables are added as noted in [72].

Indeed, if we normalize the output of (5.1) to a mean-zero unit-variance variable, and if the $\{u_i, i \geq 1\}$ are iid with finite variance, then the normalized output converges to a Gaussian distribution as d_v tends to infinity by the central limit theorem. However, due to the interaction between variable and check nodes, it does not seem easy to rigorously show how close the distributions are to the Gaussian in actual density evolution.

Using simulations, Wiberg [72] observed that message distributions for AWGN channels resemble Gaussians using a (2,3)-regular code. Our empirical results using density evolution show that for a regular graph, the output v in (5.1) as well as the output u in (5.2) can be well approximated by Gaussian densities, although u tends to be less like Gaussian, especially when its mean is near zero. Despite this difference, the approximation method that we develop in this section works very well for regular LDPC codes. From this point on, for regular LDPC codes, we assume that the variables u, v, u_i, v_j's are Gaussian.

Since a Gaussian is completely specified by its mean and variance, we need to keep only the means and variances during iterations. However, by enforcing the symmetry condition for the approximate Gaussian densities at every iteration, we can reduce the problem to a one-dimensional one, since for a Gaussian with mean m and variance σ^2, the symmetry condition reduces to $\sigma^2 = 2m$. This means that we need to keep only the mean. In fact, this not only makes the problem simpler, but also greatly improves the accuracy of the approximation.

In [16], the SNR of a Gaussian was used instead of its mean, to approximate density evolution for turbo codes. If we define the SNR of a Gaussian with mean m and variance σ^2 as m^2/σ^2, then it becomes $m/2$ if we assume the symmetry condition for the approximate messages. Therefore, the SNR becomes equivalent to the mean. However, since simulations were used in [16] due to a lack of tools for analyzing

the density evolution for turbo codes, they were unable to calculate thresholds for turbo codes with enough precision (up to two digits). Since we are concentrating only on regular and irregular LDPC codes, we can find analytic expressions for the approximate density evolution and we can calculate the approximate threshold values with an arbitrary precision. Our empirical results show it is limited to about ten digits due to the precision of the double precision numbers. Using the mean is also intuitive and physically motivated especially for irregular codes. It needs more steps to show how different SNR's are mixed for irregular codes, which can be done naturally using the mean as we will show in the next section.

We denote the means of u and v by m_u and m_v, respectively. Then (5.1) simply becomes

$$m_v^{(\ell)} \;=\; m_{u0} + (d_v - 1) m_u^{(\ell-1)}, \tag{7.1}$$

where ℓ denotes ℓ-th iteration. We omit the index i because the u_i's are iid for $1 \le i < d_v$, and have the same mean m_u. Note that $m_u^{(0)} = 0$ since the initial message from any check node is 0.

The updated mean $m_u^{(\ell)}$ at the ℓ-th iteration can be calculated by taking expectations on each side of (5.2), i.e.,

$$E\left[\tanh\frac{u^{(\ell)}}{2}\right] = E\left[\tanh\frac{v^{(\ell)}}{2}\right]^{d_c-1}, \tag{7.2}$$

where we have omitted the index j and simplified the product because the v_j's are iid. One complete iteration begins at the variable nodes and then ends at the check nodes. The variables $u^{(\ell)}$ and $v^{(\ell)}$ are Gaussian $\mathcal{N}(m_u^{(\ell)}, 2m_u^{(\ell)})$ and $\mathcal{N}(m_v^{(\ell)}, 2m_v^{(\ell)})$, respectively.

Note that the expectation $E\left[\tanh\frac{u}{2}\right]$ depends only on the mean m_u of u, since u is Gaussian with mean m_u and variance $2m_u$; i.e.,

$$E\left[\tanh\frac{u}{2}\right] \;=\; \frac{1}{\sqrt{4\pi m_u}}\int_{\mathbb{R}} \tanh\frac{u}{2} e^{-\frac{(u-m_u)^2}{4m_u}} du.$$

We define the following function $\phi(x)$ for $x \in [0, \infty)$, which will be useful and convenient for further analyses:

Definition 7.1

$$\phi(x) = \begin{cases} 1 - \frac{1}{\sqrt{4\pi x}} \int_{\mathbb{R}} \tanh \frac{u}{2} e^{-\frac{(u-x)^2}{4x}} du, & \text{if } x > 0; \\ 1, & \text{if } x = 0. \end{cases}$$

It is easy to check that $\phi(x)$ is continuous and monotonically decreasing on $[0, \infty)$, with $\phi(0) = 1$ and $\phi(\infty) = 0$. Using $\phi(x)$, we obtain the following update rule for m_u:

$$m_u^{(\ell)} = \phi^{-1}\left(1 - \left[1 - \phi\left(m_{u_0} + (d_v - 1)m_u^{(\ell-1)}\right)\right]^{d_c - 1}\right), \tag{7.3}$$

where $m_u^{(0)} = 0$ is the initial value for m_u.

The following lemma gives upper and lower bounds on $\phi(x)$.

Lemma 7.1

$$\sqrt{\frac{\pi}{x}} e^{-\frac{x}{4}} \left(1 - \frac{3}{x}\right) < \phi(x) < \sqrt{\frac{\pi}{x}} e^{-\frac{x}{4}} \left(1 + \frac{1}{7x}\right), x > 0 \tag{7.4}$$

Proof: By expanding $\frac{1}{1+e^u}$ as follows,

$$\frac{1}{1+e^u} = \begin{cases} \sum_{k=0}^{\infty} (-1)^k e^{ku}, & \text{if } u < 0; \\ \sum_{k=1}^{\infty} (-1)^{k-1} e^{-ku}, & \text{if } u > 0, \end{cases}$$

we obtain

$$\begin{aligned}\phi(x) &= \frac{1}{\sqrt{\pi x}}\int_{-\infty}^{0}\sum_{k=0}^{\infty}(-1)^k e^{ku}e^{-(u-x)^2/4x}du + \frac{1}{\sqrt{\pi x}}\int_{0}^{\infty}\sum_{k=1}^{\infty}(-1)^{k-1}e^{-ku}e^{-(u-x)^2/4x}du \\ &= \frac{2}{\sqrt{\pi x}}e^{-x/4}\int_{0}^{\infty}\sum_{k=0}^{\infty}(-1)^k e^{-(\frac{1}{2}+k)u}e^{-u^2/4x}du \\ &= 4\sum_{k=0}^{\infty}(-1)^k e^{x(k^2+k)}Q\left(\sqrt{\frac{x}{2}}(1+2k)\right),\end{aligned}$$

where $Q(x) = \frac{1}{\sqrt{2\pi}}\int_x^\infty e^{-t^2/2}dt$ is the Q-function. Using $(\frac{1}{x}-\frac{1}{x^3})\frac{1}{\sqrt{2\pi}}e^{-x^2/2} < Q(x) < \left(\frac{1}{x}\right)\frac{1}{\sqrt{2\pi}}e^{-x^2/2}$ [74], we get

$$\begin{aligned}\phi(x) &< \frac{4e^{-x/4}}{\sqrt{\pi x}}\left[\sum_{k=0}^{\infty}\frac{(-1)^k}{1+2k} + \sum_{j=1}^{\infty}\frac{2}{x(4j-1)^3}\right] \\ &< \sqrt{\frac{\pi}{x}}e^{-x/4}\left(1+\frac{1}{7x}\right),\end{aligned}$$

where we use $\sum_{k=0}^{\infty}\frac{(-1)^k}{1+2k} = \frac{\pi}{4}$ [33] and $\sum_{j=1}^{\infty}\frac{1}{(4j-1)^3} < \frac{\pi}{56}$. Similarly, we get

$$\begin{aligned}\phi(x) &> \frac{4e^{-x/4}}{\sqrt{\pi x}}\left[\sum_{k=0}^{\infty}\frac{(-1)^k}{1+2k} - \sum_{j=1}^{\infty}\frac{2}{x(4j-3)^3}\right] \\ &> \sqrt{\frac{\pi}{x}}e^{-x/4}\left(1-\frac{3}{x}\right),\end{aligned}$$

where we use $\sum_{j=1}^{\infty}\frac{1}{(4j-3)^3} < \frac{3\pi}{8}$. □

These bounds, which are tight as $x \to \infty$, will be used in analyzing the behavior of the decoder and in optimizing degree distributions in the following sections.

Although it is possible to calculate thresholds and to optimize degree distributions using the exact values of $\phi(x)$, we note that the following two approximations can be used instead to speed up the calculations without much sacrifice in accuracy. For small x, say $x < 10$, we will use the following approximation, which is very accurate

and is better than (7.4) for this range of x:

$$\phi(x) \sim e^{\alpha x^{\gamma}+\beta}, \tag{7.5}$$

where $\alpha = -0.4527, \beta = 0.0218$ and $\gamma = 0.86$ (these values were optimized to minimize the maximum absolute error between (7.5) and $\phi(x)$). We note that there should be other curves that fit $\phi(x)$ better than (7.5), but (7.5) seems to be good enough for our purposes. For large x, say $x > 10$, we use the average of the upper and lower bounds of (7.4) as an approximation for $\phi(x)$.

Table 7.1 shows approximate thresholds σ_{GA} calculated using (7.3) compared to the exact thresholds σ_{exact} from density evolution. There is about 0.3% to 1.2% error (0.03 to 0.1 dB) compared to the exact values. Tables 7.2 and 7.3 show that the Gaussian approximation is also good for Laplace channels and for BSCs, where the thresholds are calculated using the Gaussian approximation assuming the AWGN channel and then mapped to other channel domains.

Figure 7-1 shows differences between approximate and exact thresholds for various regular LDPC codes, where SNR_{norm} is defined as the distance from the Shannon limit in dB. Note that the accuracy for $(j, j+1)$ codes becomes improved as j becomes large, which can be explained because as more inputs are added at a variable node then the output becomes more Gaussian. Discretized density evolution was used to calculate exact thresholds in Table 7.1.

7.2.2 Gaussian Approximation for Irregular LDPC Codes

The previous analysis can be easily extended to irregular LDPC codes.

We again consider an ensemble of random codes with degree distributions $\lambda(x)$ and $\rho(x)$. A node will thus get iid messages from its neighbors, where each of these messages is a random mixture of different densities from neighbors with different degrees. We denote these mixtures from variable nodes and from check nodes by v and u, respectively.

We assume first that the individual output of a variable or a check node is Gaus-

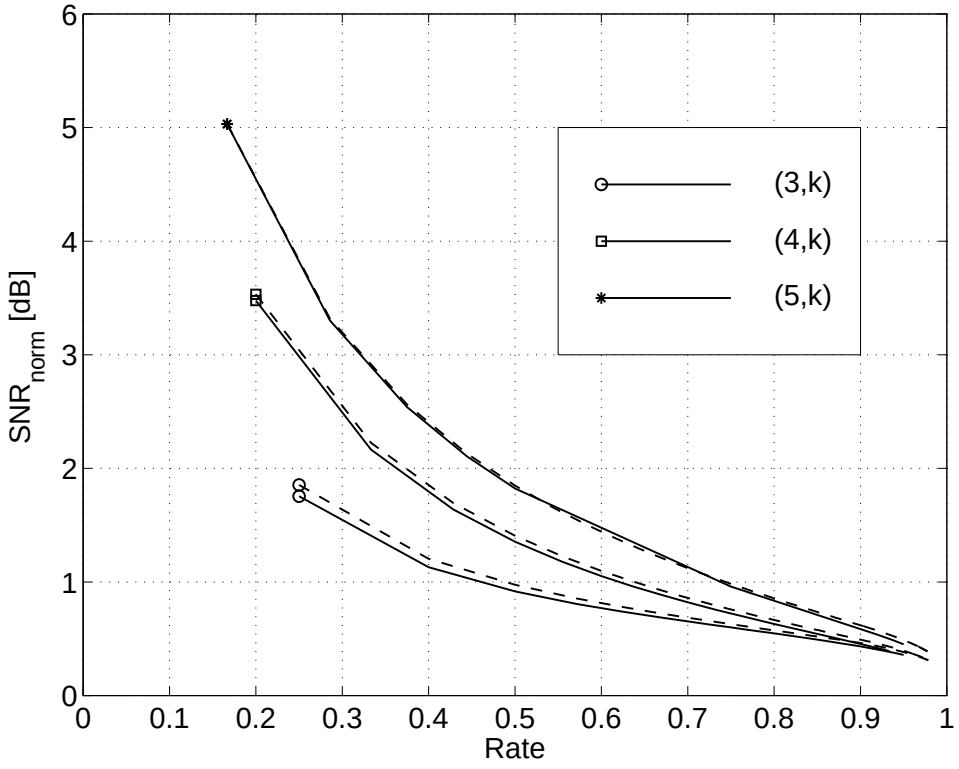


Figure 7-1: Exact (—), Gaussian approximation (- -), and reciprocal-channel approximation (- ·) thresholds for (j, k)-regular LDPC codes. Thresholds using the reciprocal-channel approximation are not distinguishable from the exact thresholds at this scale.

sian, as in the last section, based on the empirical evidence. Therefore, the mean $m_{v,i}^{(\ell)}$ of the output message of a degree-i variable node at ℓ-th iteration is given by

$$m_{v,i}^{(\ell)} = m_{u_0} + (i-1)m_u^{(\ell-1)},$$

where m_{u_0} is the mean of u_0 and $m_u^{(\ell-1)}$ is the mean of u at the $(\ell-1)$-st iteration, where u is a Gaussian mixture in general (when $\rho(x)$ is not concentrated in one degree). The variance of the output density is given by $2m_{v,i}^{(\ell)}$, as in the last section. Therefore, at the ℓ-th iteration, an incoming message v to a check node will have the following Gaussian mixture density $f_v^{(\ell)}$:

$$f_v^{(\ell)} = \sum_{i=2}^{d_l} \lambda_i \mathcal{N}(m_{v,i}^{(\ell)}, 2m_{v,i}^{(\ell)}), \tag{7.6}$$

where $m_{v,i}^{(\ell)}$ is the mean of the Gaussian output from a degree-i variable node.

Using (7.6), we get

$$E\left[\tanh\frac{v^{(\ell)}}{2}\right] = 1 - \sum_{i=2}^{d_l} \lambda_i \phi(m_{v,i}^{(\ell)}).$$

Then, we use (7.2) for a check node to calculate the mean of its output. Therefore, the mean $m_{u,j}^{(\ell)}$ of the Gaussian output message $u_j^{(\ell)}$ of a check node with degree j at ℓ-th iteration is given by

$$m_{u,j}^{(\ell)} = \phi^{-1}\left(1 - \left[1 - \sum_{i=2}^{d_l} \lambda_i \phi(m_{v,i}^{(\ell)})\right]^{j-1}\right).$$

Since $u = u_j$ with probability ρ_j for $2 \le j \le d_r$, u is also a Gaussian mixture. Since the output message of a variable node is characterized by its mean, which is the sum of the means of the incoming densities, we need to keep only the mean of the Gaussian mixture from check nodes to variable nodes. By linearly combining these means for degree-$2, \ldots, d_r$ check nodes with weights $\{\rho_i, 2 \le i \le d_r\}$, we get $m_u^{(\ell)}$ as given in

the following:

$$m_u^{(\ell)} = \sum_{j=2}^{d_r} \rho_j \phi^{-1}\left(1 - \left[1 - \sum_{i=2}^{d_l} \lambda_i \phi\left(m_{u_0} + (i-1)m_u^{(\ell-1)}\right)\right]^{j-1}\right), \tag{7.7}$$

where $m_{u_0} = 2/\sigma_n^2$ is the mean of the message from the channel, σ_n^2 is the noise variance of the AWGN channel, and the initial value of $m_u^{(0)}$ is set to 0.

For $0 < s < \infty$ and $0 \leq t < \infty$, we define $f_j(s,t)$ and $f(s,t)$ as

$$\begin{aligned} f_j(s,t) &= \phi^{-1}\left(1 - \left[1 - \sum_{i=2}^{d_l} \lambda_i \phi\left(s + (i-1)t\right)\right]^{j-1}\right); \\ f(s,t) &= \sum_{j=2}^{d_r} \rho_j f_j(s,t). \end{aligned} \tag{7.8}$$

Using this, we rewrite (7.7) as

$$t_\ell = f(s, t_{\ell-1}), \tag{7.9}$$

where $s = m_{u_0}$ and $t_\ell = m_u^{(\ell)}$. The initial value t_0 is 0. Note that since $t_1 = f(s,0) > 0$ for $s > 0$, the iteration (7.9) will always start.

Definition 7.2 *The threshold s^* is the infimum of all s in $\mathbb{R}^+$ such that $t_\ell(s)$ converges to ∞ as $\ell \to \infty$.*

The threshold in terms of noise power is therefore equal to $\frac{2}{s^*}$. Since $\phi(x)$ is monotonically decreasing on $0 \leq x < \infty$, we conclude that $f(s,t)$ is monotonically increasing on both $0 < s < \infty$ and $0 \leq t < \infty$. By finite induction, we conclude that for all $s > s^*$, $t_\ell(s) > t_\ell(s^*)$ and $t_\ell(s)$ will converge to ∞.

Lemma 7.2 *$t_\ell(s)$ will converge to ∞ iff*

$$t < f(s,t) \tag{7.10}$$

for all $t \in \mathbb{R}^+$.

Proof: Let us first assume $t < f(s,t)$, $\forall t \in \mathbb{R}^+$. Since $t_{\ell+1} > t_\ell$, t_ℓ will converge to $t_\infty \in (0,\infty]$. However, t_∞ cannot be finite, because if so then $t_\infty = f(s,t_\infty)$.

Now we assume $t' = f(s,t')$ for some $t' \in \mathbb{R}^+$. $t' \neq 0$ because $f(s,0) > 0$. Since f is continuous, there exists $t_0 > 0$ such that $t < f(s,t)$ for $\forall t \in (0,t_0)$. Now we choose the smallest $t' \in \mathbb{R}^+$ such that $t' = f(s,t')$ and $t < f(s,t)$ for $\forall t \in (0,t')$. Since $0 < f(s,0) < f(s,t') = t'$ and $f(s,t) < f(s,t') = t'$ for $\forall t \in (0,t')$, we conclude that $t_\ell < t'$ for $\forall \ell \geq 0$. $\square$

As an alternative expression to (7.8), for $0 < s < \infty$ and $0 < r \leq 1$, we define $h_i(s,r)$ and $h(s,r)$ as

$$\begin{aligned} h_i(s,r) &= \phi\left(s + (i-1)\sum_{j=2}^{d_r} \rho_j \phi^{-1}\left(1-(1-r)^{j-1}\right)\right); \\ h(s,r) &= \sum_{i=2}^{d_l} \lambda_i h_i(s,r). \end{aligned} \tag{7.11}$$

Now, (7.9) becomes equivalent to

$$r_\ell = h(s, r_{\ell-1}), \tag{7.12}$$

where $s = m_{u_0}$. The initial value r_0 is $\phi(s)$. It is easy to see that $t_\ell \to \infty$ iff $r_\ell \to 0$. Thus, the convergence condition $r_\ell(s) \to 0$ is satisfied iff

$$r > h(s,r), \forall r \in (0, \phi(s)). \tag{7.13}$$

This will be useful in optimizing $\lambda(x)$, since (7.11) is linear in the λ_i's.

Let us take an example to demonstrate how well message densities are approximated by Gaussian mixtures. The rate-1/2 code with $d_l = 20$ in Table 8.1, designed using density evolution, has a threshold 0.9669. The threshold calculated by the Gaussian approximation is 0.9473. Figure 7-2 shows two message densities from variable to check nodes when density evolution and the Gaussian approximation are used for the $d_l = 20$ code in Table 8.1. The noise power was set to be 0.006 dB below

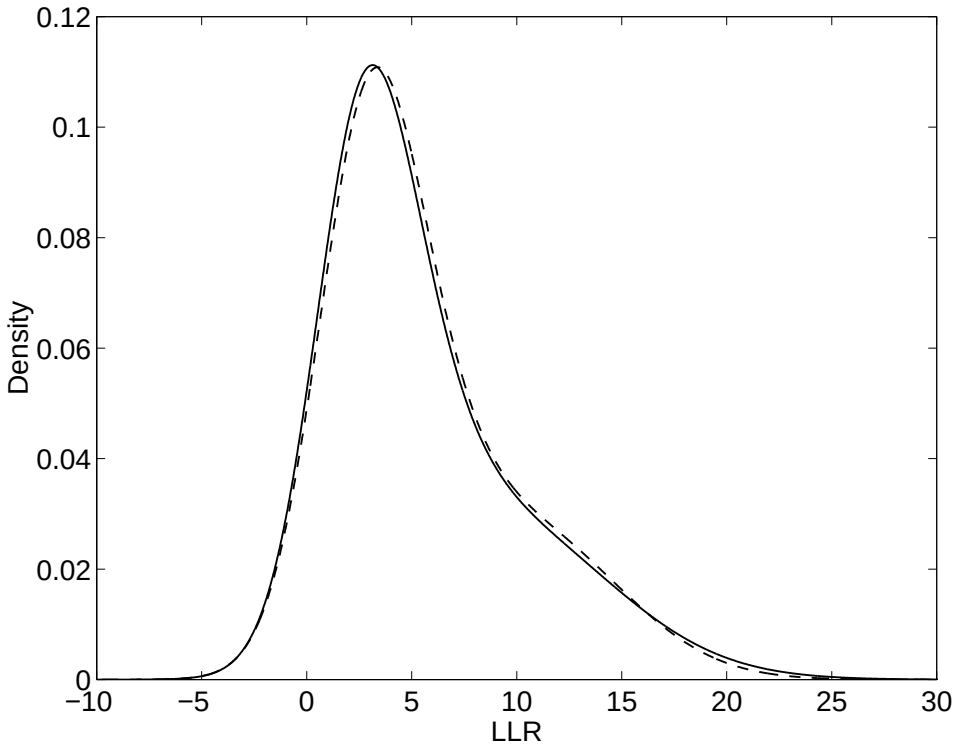


Figure 7-2: Approximate (- -) and exact (—) densities at the check node input

the threshold in each case. Surprisingly, they are very close. Figure 7-3 shows corresponding densities from check to variable nodes. In this case, the exact density has a spike at 0, which cannot be modeled well by a Gaussian approximation. Even though the two densities look very different, the fact that the two densities in the other direction are very close suggests that the approximation is working well. In fact, even though the exact density in Figure 7-3 is very different from a Gaussian, the output of a variable node will be Gaussian-like again because many iid inputs are added at a variable node. Furthermore, when the probability of error is small enough, the exact density in Figure 7-3 becomes Gaussian-like, as observed in [53]. We note that such a mismatch between the exact and approximate densities in Figure 7-3 still exists in regular cases, too, because the shape of the exact density in Figure 7-3 is due to the nature of check node computations (the tanh rule), not because $\rho(x)$ is irregular.

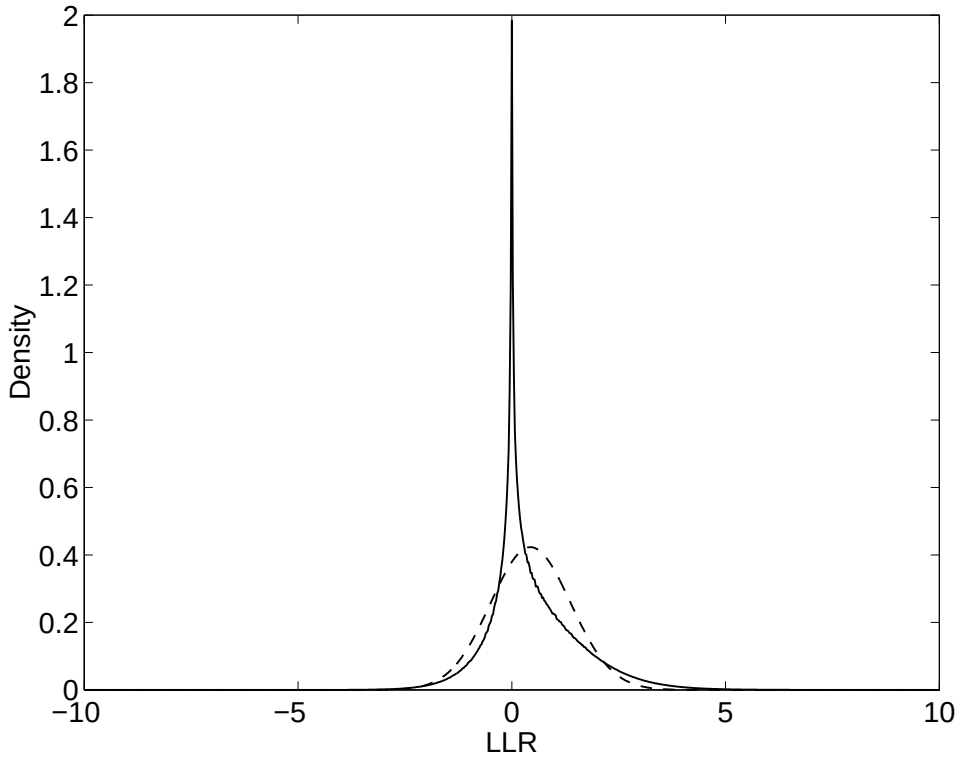


Figure 7-3: Approximate (- -) and exact (—) densities at the variable node input

7.2.3 Stability

In this section, we find the condition for the convergence of t_ℓ to ∞ in (7.9) as ℓ tends to infinity when t_0 is large enough, also known as the stability condition [53]. We also derive the rate of convergence of the probability of error using this result.

We first prove the following lemma, which gives the behavior of (7.8) when t is large (equivalently, small probability of error).

Lemma 7.3 *As $t \to \infty$, $\Delta t = f(s,t) - t$ becomes*

$$\Delta t = s + (i_1 - 2)t - 4\log\lambda_{i_1} - 4\sum_{j=2}^{d_r} \rho_j \log(j-1) + O(t^{-1}), \tag{7.14}$$

where λ_{i_1} is the first nonzero λ_i.

Proof: When t is large in (7.8), $\sum_{i=2}^{d_l} \lambda_i \phi\left(s+(i-1)t\right)$ can be simplified to

$$\sum_{i=2}^{d_l} \lambda_i \phi\left(s+(i-1)t\right) \;=\; \lambda_{i_1} \phi\left(s+(i_1-1)t\right) + O(\lambda_{i_2} \phi\left(s+(i_2-1)t\right)),$$

where λ_{i_1} and λ_{i_2} are the first and the second nonzero λ_i's, respectively. If $\lambda_k = 1$ for some $k \geq 2$, then we define $i_2 = k+1$. Using $(1+x)^n = 1 + nx + O(x^2)$, we get

$$f_j(s,t) = \phi^{-1}\left((j-1)\lambda_{i_1} \phi(s+(i_1-1)t)\left[1+O\left(\phi(s+(i_1-1)t)\right)\right]\right), \qquad (7.15)$$

where we used $\phi(s+(i_2-1)t) \ll \phi(s+(i_1-1)t)$. By taking $\phi(\cdot)$ on both sides of (7.15), we get

$$\begin{aligned} \phi(f_j(s,t)) &= (j-1)\lambda_{i_1} \phi(s+(i_1-1)t)\left[1+O\left(\phi(s+(i_1-1)t)\right)\right] \\ &= \left(1+O\left(t^{-1}\right)\right) \frac{1}{\sqrt{s+(i_1-1)t}} e^{-(s+(i_1-1)t - 4\log(j-1)\lambda_{i_1})/4}, \end{aligned}$$

where we used $\phi(s+(i_1-1)t) \ll t^{-1}$ and Lemma 7.1. By using Lemma 7.1 again, we get

$$\phi(f_j(s,t)) \;=\; \phi\left(s+(i_1-1)t - 4\log(j-1)\lambda_{i_1} + O(t^{-1})\right).$$

Finally by taking $\phi^{-1}(\cdot)$ and by using (7.8), we get the result. □

If $i_1 > 2$, then t_ℓ always converges to infinity regardless of s and $\rho(x)$ as ℓ tends to infinity if t_0 is large enough. Otherwise, Δt becomes a constant as t becomes large. For this case, using $s = 2/\sigma_n^2$, we get

$$\Delta t = 4\log\frac{\lambda_2^*}{\lambda_2} + O(t^{-1}),$$

where $\lambda_2^* = e^{1/2\sigma_n^2} / \prod_{j=2}^{d_r} (j-1)^{\rho_j}$. The following theorem summarizes these two results.

Theorem 7.4 *If $\lambda_2 < \lambda_2^*$, then t_ℓ will converge to infinity as ℓ tends to infinity if t_0*

is large enough. If $\lambda_2 > \lambda_2^*$, then t_ℓ cannot converge to infinity regardless of its initial value.

Note that this condition is similar to the stability condition $\lambda_2 < e^{1/2\sigma_n^2} / \sum_{j=2}^{d_r} \rho_j (j-1)$ of [53], except that $\sum_{j=2}^{d_r} \rho_j (j-1)$ is replaced by $\prod_{j=2}^{d_r} (j-1)^{\rho_j}$. Using Jensen's inequality, we conclude that $\prod_{j=2}^{d_r} (j-1)^{\rho_j} \leq \sum_{j=2}^{d_r} \rho_j (j-1)$ for all $\rho(x)$, with equality iff $\rho_k = 1$ for some $k \geq 2$. Therefore, the approximate stability condition is looser than the original, which implies that stable degree distributions under the Gaussian approximation may not be stable under density evolution. However, stability under density evolution guarantees stability under the Gaussian approximation. In any case, these two are the same if $\rho_k = 1$ for some $k \geq 2$, and are very close if $\rho(x)$ is concentrated at two consecutive degrees, which is the case for most good degree distributions found so far.

If $0 < \lambda_2 < \lambda_2^*$, then for large t_ℓ we can express t_ℓ as

$$t_\ell \approx c + 4\ell \log \frac{\lambda_2^*}{\lambda_2}, \tag{7.16}$$

where c is a constant. The probability of error P_ℓ at the ℓ-th iteration is given by

$$P_\ell = \sum_{i=2}^{d_l} \lambda_i' Q\left(\sqrt{\frac{s + it_\ell}{2}} \right), \tag{7.17}$$

where $\lambda_i' = \frac{\lambda_i / i}{\sum_{j=2}^{d_l} \lambda_j / j}$ is the fraction of degree-i nodes. When t_ℓ is large, we get the following approximation for the probability of error.

Lemma 7.5 *When t_ℓ is large, P_ℓ is approximated by*

$$P_\ell \approx \begin{cases} \frac{a}{\sqrt{b+\ell}} \left(\frac{\lambda_2}{\lambda_2^*} \right)^{2\ell}, & \text{if } 0 < \lambda_2 < \lambda_2^*; \\ \frac{d}{(i_1-1)^{\ell/2}} e^{-f(i_1-1)^\ell}, & \text{if } \lambda_2 = 0, \end{cases}$$

where a, b, d, f are positive constants that depend on s and the degree distributions.

Proof: When t_ℓ is large, the first nonzero term in (7.17) becomes dominant. When

$0 < \lambda_2 < \lambda_2^*$, P_ℓ becomes

$$\begin{aligned} P_\ell &\approx \lambda_2' Q\left(\sqrt{\frac{s+2t_\ell}{2}}\right) \\ &\approx \frac{a}{\sqrt{b+\ell}}\left(\frac{\lambda_2}{\lambda_2^*}\right)^{2\ell}, \end{aligned}$$

where a and b are constants that depend on the degree distributions and s, and we have used $Q(\sqrt{\frac{x}{2}}) = (1+O(x^{-1}))\frac{1}{\sqrt{\pi x}}e^{-x/4}$ and (7.16). If $\lambda_2 = 0$, then t_ℓ becomes approximately $d'(i_1-1)^\ell$, where d' is a positive constant that depends on the degree distributions and s. Using this and the above approximation for the Q-function, the result follows. □

This implies that when $0 < \lambda_2 < \lambda_2^*$ and P_ℓ is small, the error probability P_ℓ will decrease by a factor of $(\lambda_2/\lambda_2^*)^2$ at each iteration, which matches empirical observations using density evolution very well.

7.2.4 Optimization of Degree Distributions using Gaussian Approximation

Assuming that $\lambda(x)$ and σ_n are given, we can optimize $\rho(x)$ by maximizing the rate of the code. The constraints are (1) normalization: $\rho(1) = 1$; and (2) the inequality constraint (7.10). Since this is a linear program, it can be solved efficiently. Alternatively, however, we can maximize (7.14) ignoring the $O(t^{-1})$ term. As shown in the following theorem, the $\rho(x)$ optimized in this way has the form $\rho(x) = \rho x^{k-1} + (1-\rho)x^k$ for some $k \geq 2$ and $0 < \rho \leq 1$.

Theorem 7.6 *A concentrated degree distribution, $\rho(x) = \rho x^{k-1} + (1-\rho)x^k$ for some $k \geq 2$ and $0 < \rho \leq 1$, minimizes $\sum_{j=2}^{d_r} \rho_j \log(j-1)$ subject to $\rho(1) = 1$ and $\int_0^1 \rho(x) = c$, where c is a constant.*

Proof: The dual linear program to the given problem is:

$$\begin{aligned} \max \quad & y_1 + cy_2 \\ \text{subject to} \quad & y_1 + \tfrac{y_2}{j} \leq \log(j-1), 2 \leq j \leq d_r \end{aligned}$$

Suppose we choose an integer k such that

$$\frac{1}{k+1} < c \leq \frac{1}{k}.$$

We claim that the following are feasible solutions to the primal and dual problems of equal cost, which are optimal solutions to both primal and dual problems by the weak duality theorem [5]:

$$\begin{aligned} \rho_k &= k(k+1)c - k; \\ \rho_{k+1} &= 1 - \rho_k; \\ y_1 &= (k+1)\log k - k\log(k-1); \\ y_2 &= k(k+1)\log\frac{k-1}{k}. \end{aligned}$$

To check whether the constraints of the dual problem are satisfied, we first observe that $y_1 + \frac{y_2}{k} = \log(k-1)$, and since

$$\begin{aligned} \left(\log j - \frac{y_2}{j+1}\right) &- \left(\log(j-1) - \frac{y_2}{j}\right) \\ &= \frac{1}{j(j+1)}\left[k(k+1)\log\frac{k-1}{k} - j(j+1)\log\frac{j-1}{j}\right] \\ &\begin{cases} < 0, & \text{if } j < k; \\ = 0, & \text{if } j = k; \\ > 0, & \text{if } j > k, \end{cases} \end{aligned}$$

we conclude that $y_1 + \frac{y_2}{j} \leq \log(j-1)$, $j \geq 2$, by finite induction. □

Note that this solution also maximizes (7.14) without the $O(t^{-1})$ term, subject to the rate constraint. Experiments show that this concentrated form of $\rho(x)$ is not much

different from the optimized degree distributions using linear programming, and that there is no noticeable performance degradation for the cases we have examined so far. Thus, we will use only concentrated $\rho(x)$'s.

The optimization of $\lambda(x)$ can be done similarly to the optimization of $\rho(x)$, i.e., assuming $\rho(x)$ and σ_n are given, we optimize $\lambda(x)$ by maximizing the rate $r = 1 - \frac{\int_0^1 \rho(x)}{\int_0^1 \lambda(x)}$. The constraints are (1) normalization: $\lambda(1) = 1$; and (2) the inequality constraint (7.13). This can be also done using linear programming.

Figure 7-4 shows $\{h_i(s,r) - r\}$ for $i = 2, \ldots, 20$ using the degree distribution $\rho(x)$ of the $d_l = 20$ code in Table 8.1, where the noise level is equal to the Gaussian approximation threshold of the code. In this case, where $\rho(x)$ and the noise power are given, the optimization of $\lambda(x)$ may be performed by maximizing the rate of the code subject to the normalization constraint $\lambda(1) = 1$ and the inequality constraints $\lambda_i \geq 0$, $2 \leq i \leq d_l$, and $\sum_{i=2}^{d_l} \lambda_i (h_i(s,r) - r) < 0$, $r \in (0, \phi(s))$. Figures 7-4 and 7-5 show how each curve in $\{h_i(s,r) - r\}$ for $i = 2, \ldots, 20$, is combined to produce the curve $\{h(s,r) - r\}$, where the degree distribution $\lambda(x)$ of the $d_l = 20$ code in Table 8.1 is used. This curve is always below 0 and barely touches 0 at one point. If it hits 0, then the point r^* where the curve hits 0 becomes a fixed point, and the error probability cannot be decreased further. Note that, since the slope of the curve of $\{h_2(s,r) - r\}$ near $r = 0$ is different from those for $i > 2$, the stability condition in Theorem 7.4 is described by only λ_2.

Although we have shown that concentrated $\rho(x)$ are good enough, we may still want to visualize how the optimization of $\rho(x)$ is done. Figure 7-6 shows $\{f_i(s,t) - t\}$ for $i = 2, \ldots, 10$ using $\rho(x)$ of the $d_l = 20$ code in Table 8.1, where the noise level is equal to the Gaussian approximation threshold of the code. In this case, where $\lambda(x)$ and the noise power are given, the optimization of $\rho(x)$ may be performed by maximizing the rate of the code subject to the normalization constraint $\rho(1) = 1$ and the inequality constraints $\rho_i \geq 0$, $2 \leq i \leq d_r$, and $\sum_{i=2}^{d_r} \rho_i (f_i(s,t) - t) > 0$, $t \in (0, \infty)$. Figures 7-6 and 7-7 show how two curves corresponding to $i = 8$ and 9 in $\{f_i(s,t) - t\}$ are combined to produce the curve $\{f(s,t) - t\}$, where $\rho(x)$ of the $d_l = 20$ code in Table 8.1 is used. This curve is always above 0 and barely touches 0 at one point.

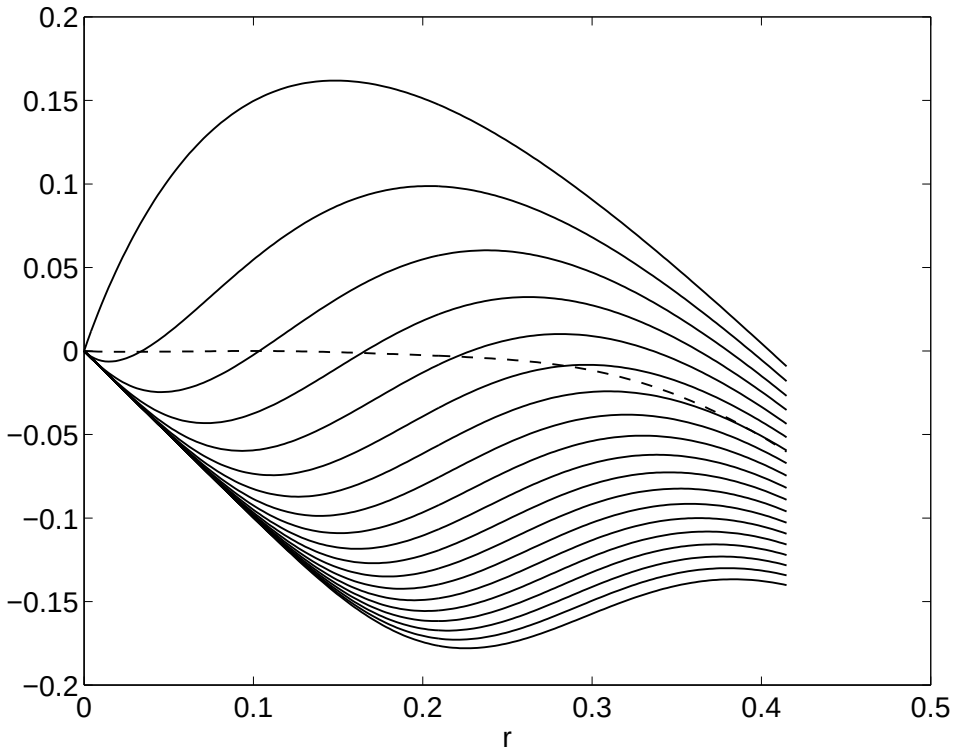


Figure 7-4: $\{h_i(s,r) - r\}$ for $i = 2, \ldots, 20$ (top to bottom) and $\{h(s,r) - r\}$ (- -)

If it hits 0, then the point t^* where the curve hits 0 becomes a fixed point, and the error probability cannot be decreased further.

Figures 7-8 and 7-9 show thresholds in $\mathrm{SNR}_{\mathrm{norm}}$ for some codes of rates 0.1 to 1 optimized using both density evolution and the Gaussian approximation when the maximum variable degree d_l is 10 or 20, respectively. They show that the difference between the exact and approximate thresholds is small (less than 0.3 dB) for codes designed using the Gaussian approximation. It also shows that the optimization based on the Gaussian approximation is good especially when d_l is low and when the rate is high. For example, for rates higher than 0.5, the difference is less than 0.02 dB for $d_l = 10$ and less than 0.1 dB for $d_l = 20$.

7.2.5 Fixed Points

We have seen that the Gaussian approximation is useful in calculating thresholds and optimizing degree distributions. However, it is not clear how accurate the approxi-

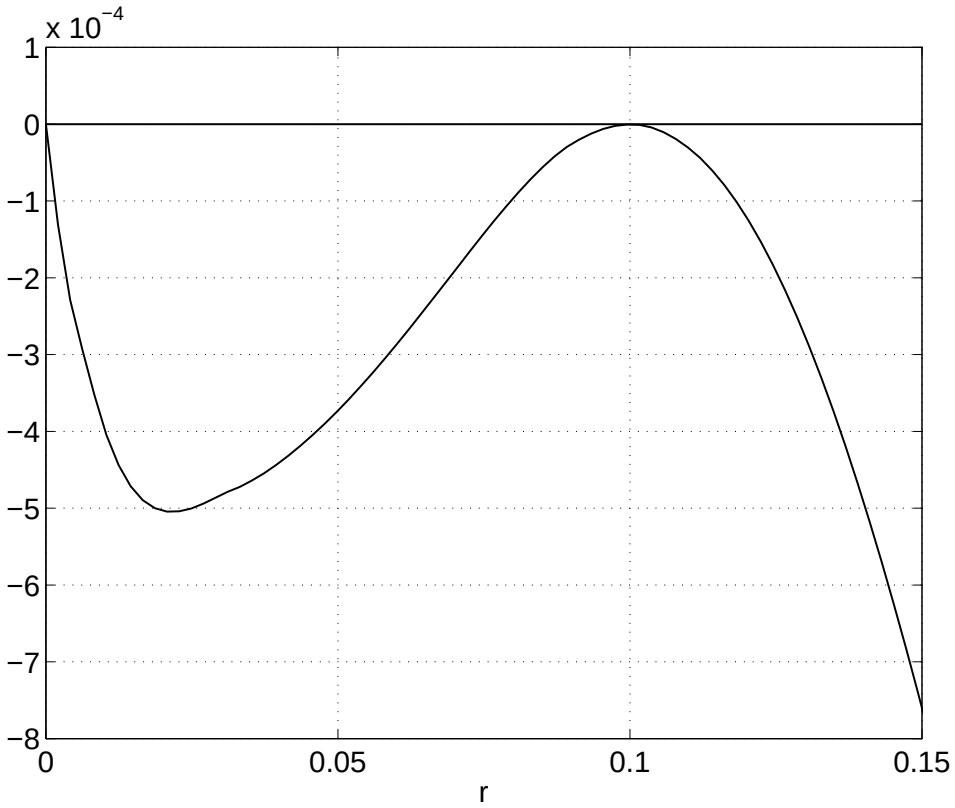


Figure 7-5: $\{h(s,r) - r\}$ as a function of r magnified

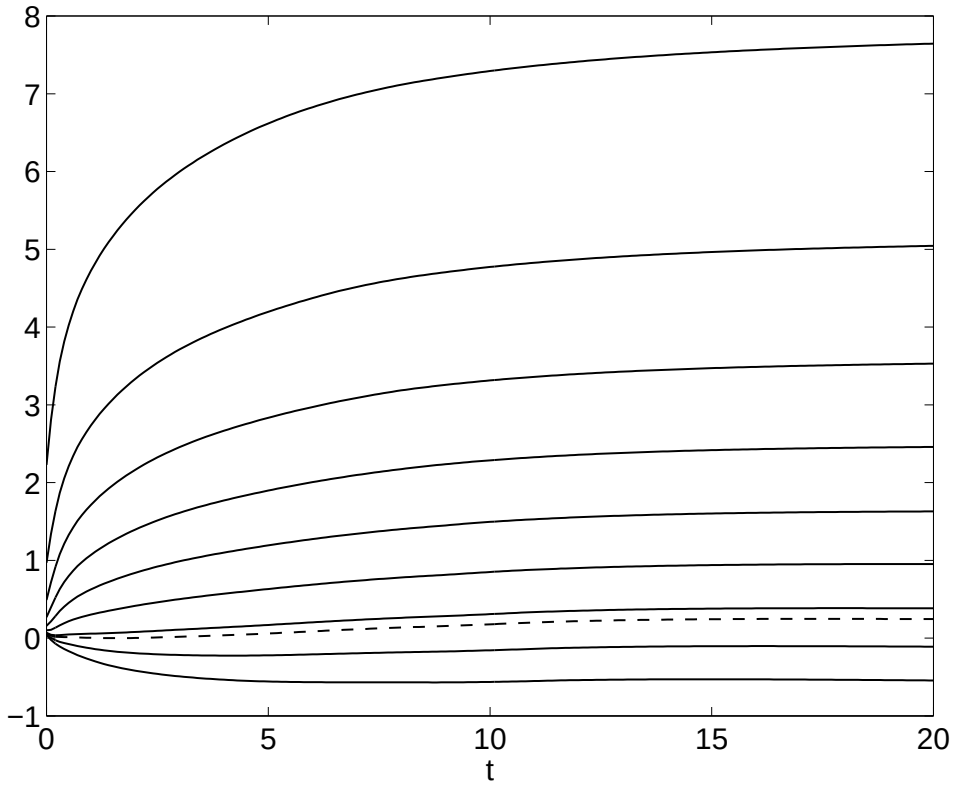


Figure 7-6: $\{f_i(s,t) - t\}$ for $i = 2, \ldots, 10$ (top to bottom) and $\{f(s,t) - t\}$ (- -)

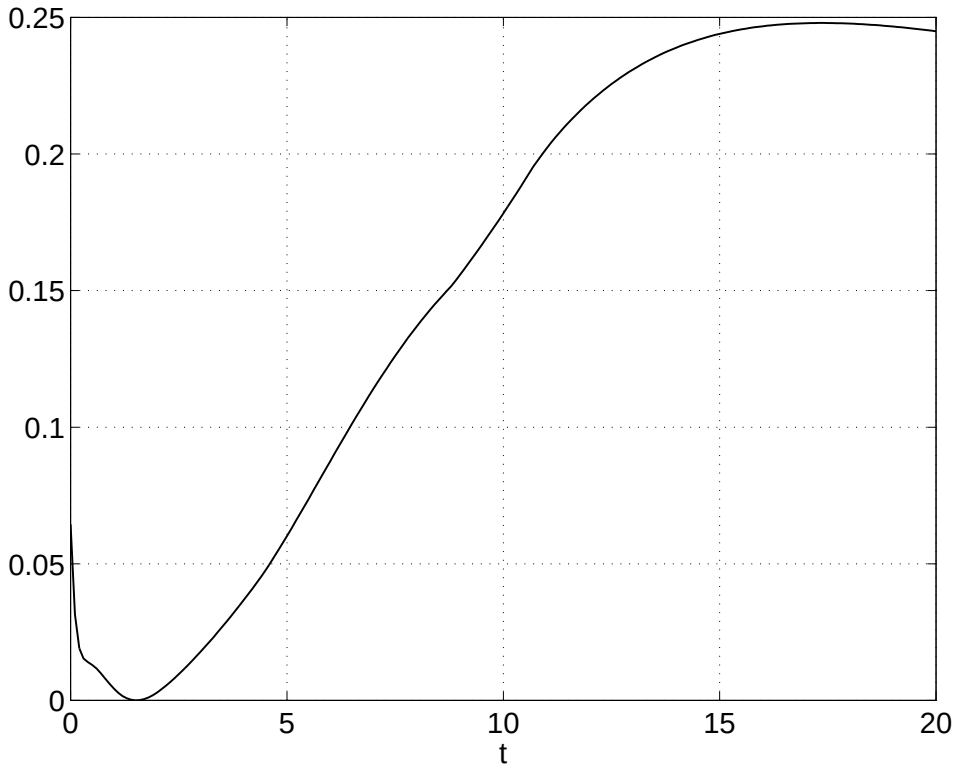


Figure 7-7: $\{f(s,t) - t\}$ as a function of t magnified

mation is by just looking at a single number, namely a threshold. In this section, we show how accurately the Gaussian approximation can predict the locations of potential fixed points of density evolution, which could show more aspects of the approximation than thresholds.

First observe that in Figures 7-5 and 7-7, there are slow and fast phases, where the width of the gap between the curve and the horizontal axis determines the iteration speed. This explains why there are slow and fast phases in sum-product decoding, as noted in [53].

Figure 7-10 shows the decrease in the probability of error as a function of the probability of error for the $d_l = 20$ code in Table 8.1 using density evolution and the Gaussian approximation, where the noise power is set to be 0.006 dB below the threshold of each case. It is interesting to observe that there are three estimated fixed points under density evolution (including zero probability of error) but only two under the Gaussian approximation. As we increase the noise power, 2×10^{-2} (probability

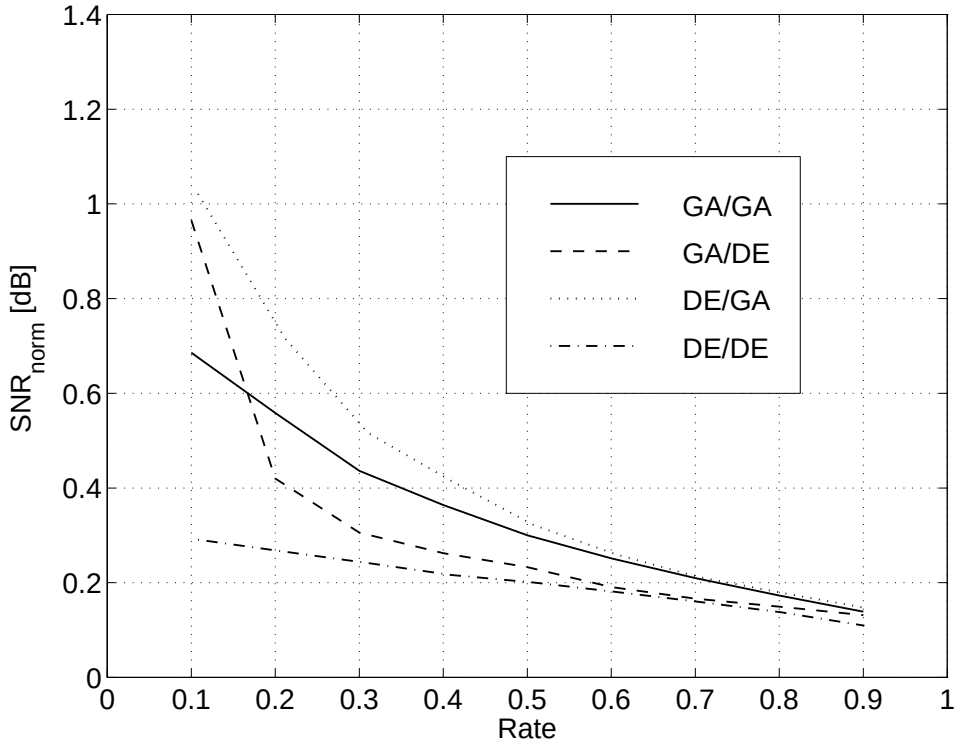


Figure 7-8: Performance of various codes, designed/evaluated using GA/GA, GA/DE, DE/GA, or DE/DE when $d_l = 10$

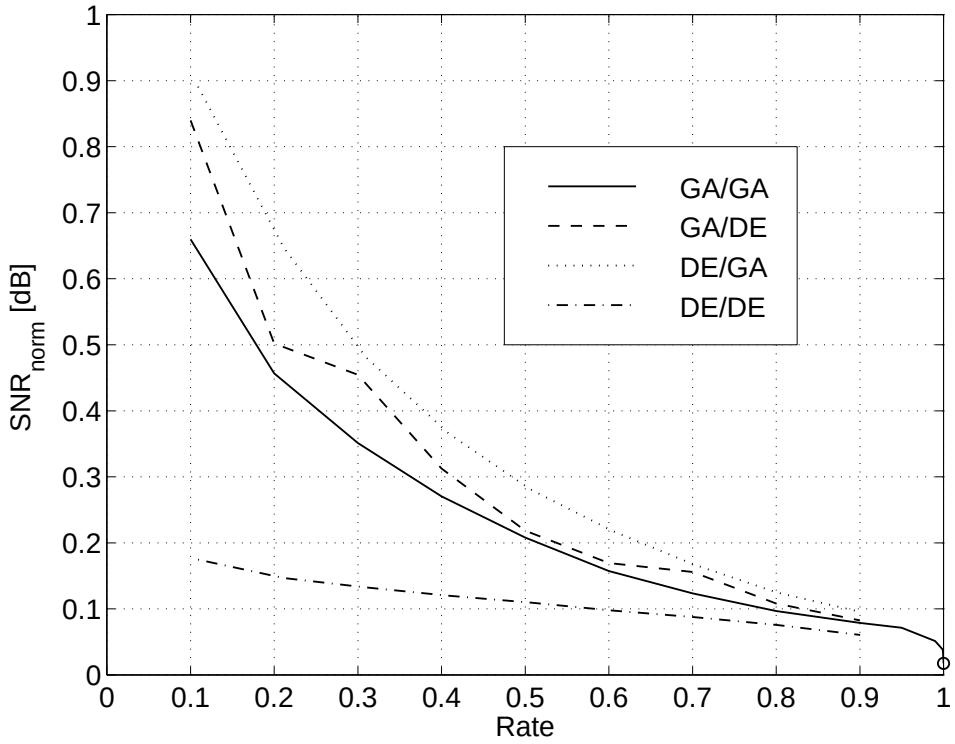


Figure 7-9: Performance of various codes, designed/evaluated using GA/GA, GA/DE, DE/GA, or DE/DE when $d_l = 20$

of error) becomes a fixed point under both the Gaussian approximation and density evolution. As the noise power is increased further, 9×10^{-2} becomes another fixed point under density evolution.

Figure 7-11 shows the decrease in the probability of error as a function of the probability of error for the following $d_l = 7$ code (rate $= 1/2$) using density evolution and the Gaussian approximation, where the noise power is set to be 0.006 dB below the threshold of each case.

$$\begin{aligned} \lambda(x) &= 0.30780x + 0.27287x^2 + 0.41933x^6 \\ \rho(x) &= 0.4x^5 + 0.6x^6 \end{aligned} \tag{7.18}$$

In this case, the number of potential fixed points is estimated correctly using the Gaussian approximation, and the two curves are quite close. In this case, the Gaussian approximation not only calculates thresholds accurately, but also tracks the exact

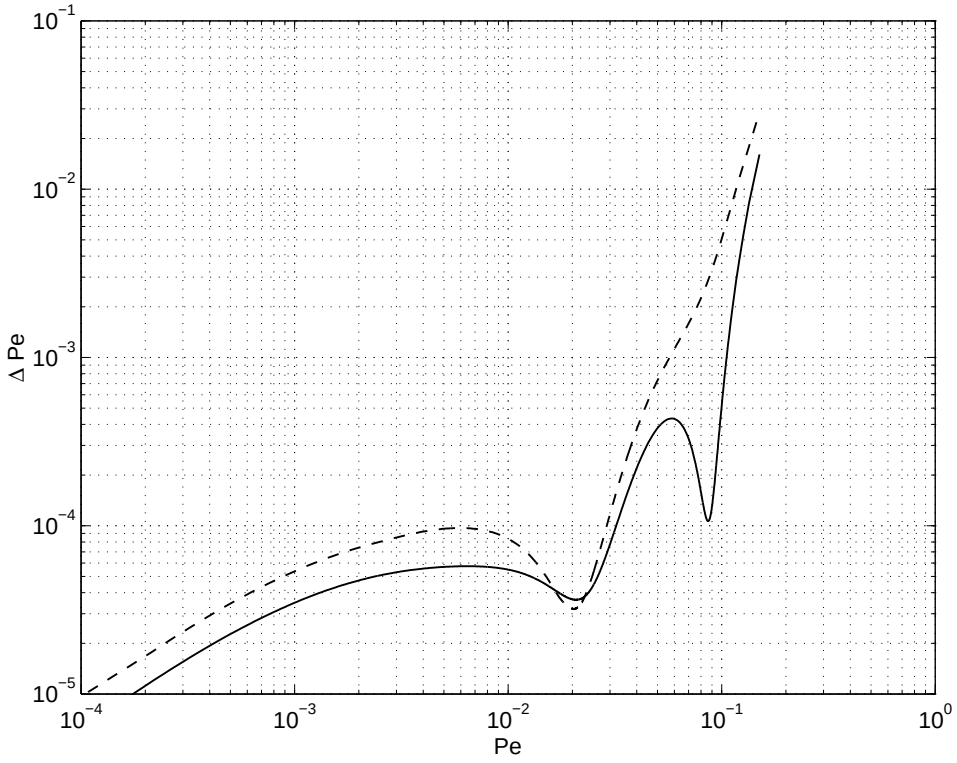


Figure 7-10: The probability of error vs. decrease in the probability of error of the $d_l = 20$ code in Table 8.1 using density evolution (—) and the Gaussian approximation (- -)

probability of error reasonably well.

We note that, from empirical observations, as the maximum variable degree increases, the number of fixed points often tends to increase. Since the Gaussian approximation is not good at estimating the number of fixed points exactly when the maximum variable degree is large, it becomes less useful in this case. However, when the maximum variable degree is small, the Gaussian approximation works reasonably well. This seems to contradict our previous observations for regular codes, where the accuracy of the Gaussian approximation for $(j, j+1)$ codes improves as j increases. This seems to suggest that, as the maximum variable degree of the code increases, the Gaussian approximation becomes less accurate because the irregularity of the code increases, even though the fraction of high-degree variable nodes increases.

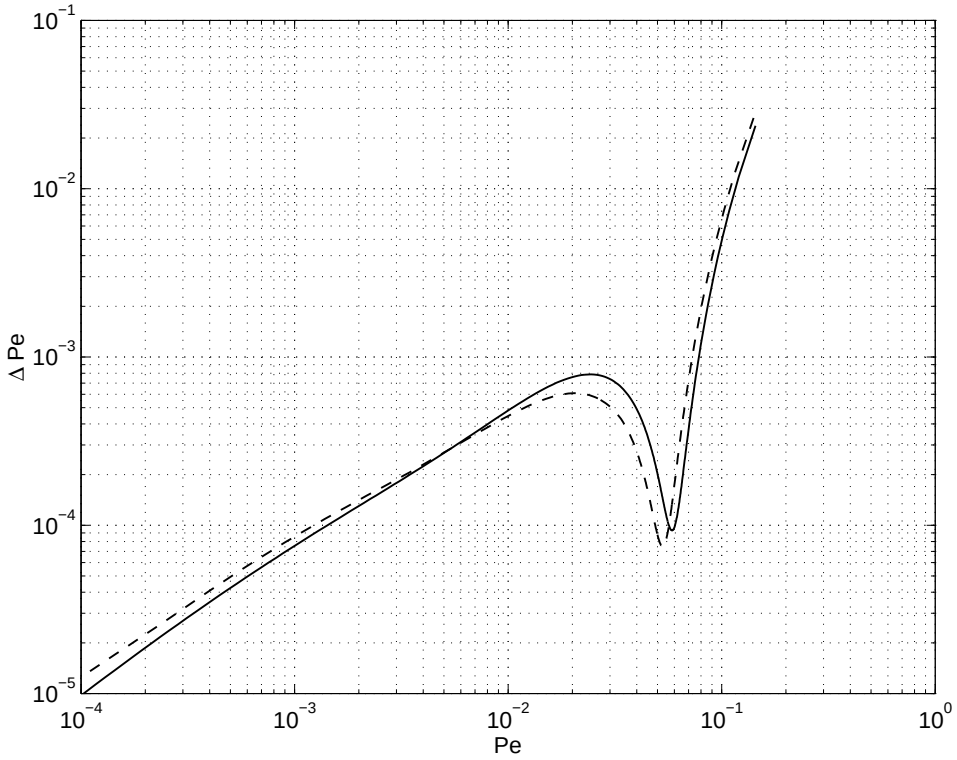


Figure 7-11: The probability of error vs. decrease in the probability of error for the $d_l = 7$ code in (7.18) using density evolution (—) and the Gaussian approximation (- -)

7.3 Gaussian-Capacity Approximation

The Gaussian approximation for check nodes in (7.2) is not unique. In fact, using a monotonically increasing function $\zeta(\cdot) : \mathbb{R} \to \mathbb{R}$, we can generate various approximate update rules for (5.2), i.e.,

$$E\left[\zeta\left(\tanh\frac{u}{2}\right)\right] = E\left[\zeta\left(\prod_{j=1}^{d_c-1}\tanh\frac{v_j}{2}\right)\right], \tag{7.19}$$

where $v_j, i = 1, \ldots, d_c - 1$, are the incoming LLRs from $d_c - 1$ neighbors of the check node and u is the message sent to the remaining neighbor, which is assumed to be a Gaussian.

Motivated by the iterative normalization in Section 5.2.3, we can treat the output message u as the output of a normalized channel whose input is the value of the hidden state carrying the message u and the output is the set of observations made in the subtree below the hidden state. A question naturally arises: what is the capacity of the channel? This is simply given by the following equation using (4.3):

$$C = 1 - E_u\left[\log_2\left(1 + e^{-u}\right)\right],$$

where u is the output message of the check node.

Using $\frac{2}{1+e^{-u}} = 1 + \tanh\frac{u}{2}$, we may rewrite this as

$$C = E_u\left[\log_2\left(1 + \tanh\frac{u}{2}\right)\right].$$

Therefore, using (5.2), we get

$$C = E_{v_1,\ldots,v_{d_c-1}}\left[\log_2\left(1 + \prod_{j=1}^{d_c-1}\tanh\frac{v_j}{2}\right)\right]. \tag{7.20}$$

Thus, we can find the mean m_u of u by setting the above two capacities equal assuming u is a Gaussian with mean m and variance $2m$. We call this "Gaussian-capacity" approximation, since at every check node output, we are replacing the

normalized channel of the check node by an AWGN channel with the same capacity.

We note that there are some difficulties in calculating (7.20), because we need either density evolution or multidimensional integral for accurately computing (7.20), but they are computationally intensive. Furthermore, they are not easy for analysis. Note that in the Gaussian approximation, many useful analyses were possible because the products of tanh's could be done outside the expectation in (7.2), which is not possible for the Gaussian-capacity approximation.

If we perform the Gaussian-capacity approximation at check nodes pairwise, then we need only a two-dimensional integral or a pairwise density evolution. We call this "pairwise Gaussian-capacity" approximation.

The difference between the Gaussian approximation in Section 7.2 and the Gaussian-capacity approximation is that $\zeta(x) = x$ for the Gaussian approximation and $\zeta(x) = \log_2(1+x)$ for the Gaussian-capacity approximation. It is not clear just from this how much difference this will make in terms of accuracy, although both approximations are developed based on some physical motivations.

In Table 7.4, we compare threshold values calculated by Gaussian-capacity and pairwise Gaussian-capacity approximations. Strangely, the pairwise Gaussian-capacity approximation performs better than the Gaussian-capacity approximation; they all perform better than the Gaussian approximation for all codes in Table 7.4. However, on average, they are still worse than the reciprocal-channel approximation, a very accurate approximation we will develop in the next section.

By using (4.3) for variable nodes we get the following:

$$C = E_v \left[\log_2 \left(1 + e^{-v} \right) \right],$$

where v is the output message of the variable node. Using (5.1), we get

$$C = E_{u_0,\dots,u_{d_v-1}} \left[\log_2 \left(1 + \prod_{i=0}^{d_v-1} e^{-u_i} \right) \right],$$

where $u_i, i = 1, \dots, d_v - 1$, are the incoming messages from the neighbors of the

Table 7.4: Exact and approximate (Gaussian-capacity approximation σ_{GCA}, pairwise Gaussian-capacity approximation σ_{pGCA}, and reciprocal-channel approximation σ_{RCA}) threshold values for various (j,k)-regular LDPC codes for the binary-input AWGN channel and sum-product decoding. All threshold values are rounded down.

j	k	rate	σ_{exact}	σ_{GCA}	error[dB]	σ_{pGCA}	error[dB]	σ_{RCA}	error[dB]
3	6	0.5	0.8809	0.8783	0.03	0.8794	0.014	0.8808	0.001
4	8	0.5	0.8376	0.8354	0.02	0.8376	0.000	0.8383	0.007
5	10	0.5	0.7936	0.7918	0.02	0.7946	0.011	0.7948	0.014
3	5	0.4	1.0093	1.0063	0.03	1.0079	0.012	1.0091	0.001
4	6	1/3	1.0109	1.0084	0.02	1.0116	0.006	1.0116	0.006
3	4	0.25	1.2667	1.2635	0.02	1.2655	0.008	1.2658	0.006
4	10	0.6	0.7481	0.7462	0.02	0.7478	0.003	0.7485	0.005
3	9	2/3	0.7082	0.7065	0.02	0.7068	0.017	0.7079	0.003
3	12	0.75	0.6320	0.6308	0.02	0.6308	0.016	0.6316	0.006

variable node except for the check node that gets the message v, and u_0 is the initial message. Although this seems to be nicely formulated, we doubt the usefulness of it, since (7.1) provided a much simpler approximation (in fact, exact for Gaussian inputs).

Another approximation method based on the stability function instead of the channel capacity is possible, since

$$\begin{aligned} S &= E_u[e^{-u/2}] \\ &= E_u\left[\sqrt{\frac{1-\tanh\frac{u}{2}}{1+\tanh\frac{u}{2}}}\right]. \end{aligned}$$

However, it turns out this is not as accurate as the Gaussian-capacity approximation, as one can guess from the results in Section 6.3 where it was shown that equal-stability curves are not as good as equal-capacity curves for mapping threshold values between different channels.

We note that the Gaussian-capacity approximation is based on the same principle as ten Brink's mutual information transfer chart [64] for parallel concatenated convolutional codes. In [64], a Gaussian approximation method based on approximating the extrinsic information from a constituent code as a Gaussian was used, so that

mutual information is the same, which was computed using a numerical integral. The mutual information is measured between a Bernoulli(1/2) source and the extrinsic information. From this Gaussian, iid inputs to the other constituent code are generated and these steps are repeated.

In [64], however, no rigorous validation or comparison was made as to why mutual information was chosen instead of other physical quantities such as the mean or the SNR of the extrinsic information. As in the Gaussian-capacity approximation for LDPC codes, this can be interpreted as approximating normalized channels by AWGN channels at each iteration, since density evolution for turbo codes [56] has essentially the same property as density evolution for LDPC codes.

Since density evolution is computationally intensive for turbo codes, Monte-Carlo simulations were used in [64] to estimate the density for extrinsic information. Compared to this, we can get arbitrary good accuracy in calculating approximate thresholds of LDPC codes using the Gaussian-capacity or pairwise Gaussian-capacity approximations, because we can use either discretized density evolution or a multidimensional numerical integral.

7.4 Reciprocal-Channel Approximation

Except for the BEC, there is no known exact one-dimensional representation of density evolution for arbitrary channels. However, as we observe from the erasure-channel and the Gaussian approximations, it seems that there are many ways of approximating density evolution reasonably well. In this section, we develop a very accurate approximation method for regular codes, which is motivated by the observations in Section 6.3. We assume (d_v, d_c)-regular codes in this section.

We first define the following mapping.

Definition 7.3 *For a given channel X, a reciprocal-channel mapping $\psi_X(\cdot) : \mathcal{X} \to \mathcal{X}$ is defined as*

$$\psi_X(x) = C_X^{-1}(1 - C_X(x)),$$

where $C_X(\cdot)$ is the capacity function and $x \in \mathcal{X}$ is the channel parameter of X.

Note that the reciprocal-channel mapping is self-inverse. Using this definition, we define the reciprocal channel as the following.

Definition 7.4 *The reciprocal channel of a channel X with a parameter x is the same channel with the parameter $\psi_X(x)$.*

We define the reciprocal-channel approximation as a one-dimensional approximation to density evolution using reciprocal-channel mappings, where all approximate messages are real numbers. To do this, we first find a mapping that maps the initial message density p_0 of a channel X to a real number a_0. It would be desirable to choose such a mapping that faithfully represent density evolution. We then find a symmetric computation rule $\tau(\cdot,\cdot)$ that maps $\mathbb{R}^2$ to $\mathbb{R}$. For a degree-d_v variable node, this function is applied repeatedly to compute the d_v-th output message b of the node, i.e.,

$$b = \tau(a_0, \tau(a_1, \tau(\dots, a_{d_v-1})\dots),$$

where $a_1, \dots, a_{d_v-1}$ are outputs of check nodes. Abusing notation, we denote the resulting mapping as $\tau(\cdot, \dots, \cdot)$.

For check nodes, we define the computation rule using the reciprocal channel mapping $\psi_{\mathrm{X}}(\cdot)$, i.e.,

$$a' = \psi_{\mathrm{X}}\left(\tau(\psi_{\mathrm{X}}(b_1), \psi_{\mathrm{X}}(b_2), \dots, \psi_{\mathrm{X}}(b_{d_c-1}))\right),$$

where the b_j's are inputs and a' is the d_c-th output of the check node.

We note that this approximation becomes exact for the BEC;

Lemma 7.7 *The reciprocal-channel approximation is exact for the BEC if the initial message is the erasure probability and $\tau(a_1, a_2) = a_1 a_2$.*

For AWGN channels, we define the mean of a LLR message as a one-dimensional

message and use $\tau(a_1, a_2) = a_1 + a_2$. In this case, we define $\psi(\cdot)$ as the following:

$$\psi(m) = \tilde{C}^{-1}_{\text{AWGN}}(1 - \tilde{C}_{\text{AWGN}}(m)),$$

where $\tilde{C}_{\text{AWGN}}(\cdot)$ is the capacity function of the AWGN channel as a function of the mean of the initial message, i.e.,

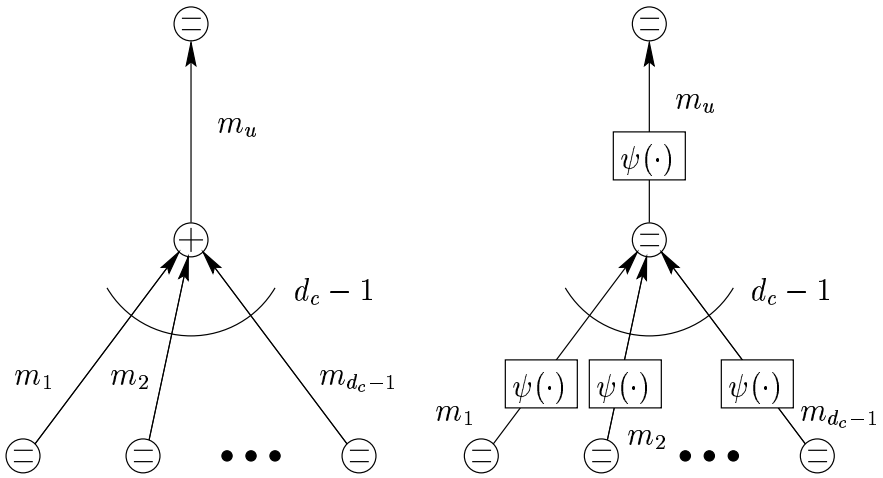
$$\tilde{C}_{\text{AWGN}}(m) = C_{\text{AWGN}}(2/m).$$

Figure 7-12 shows how computation is done at a check node. Note that the reciprocal channel mapping $\psi(\cdot)$ effectively dualizes the parity check code. Figure 7-13 shows how the reciprocal-channel approximation is computed on three equivalent graphs, i.e., (a) an original LDPC graph, (b) a graph with parity checks converted to repetition codes using $\psi(\cdot)$, and (c) a dual LDPC code with reciprocal-channel mappings attached outside the code. Since the dual LDPC code has rate equal to one minus the rate of the original code, it seems to validate the use of the reciprocal-channel mapping. However, we are not ready to claim that there is a more fundamental relationship.

We note that (c) in Figure 7-13 has states (full edges between left and right constraint nodes) initialized to infinity (AWGN) or zero (BEC) as opposed to zero (AWGN) or one (BEC) on the original graph as a result of the reciprocal-channel mappings, which is essential for the computation to start.

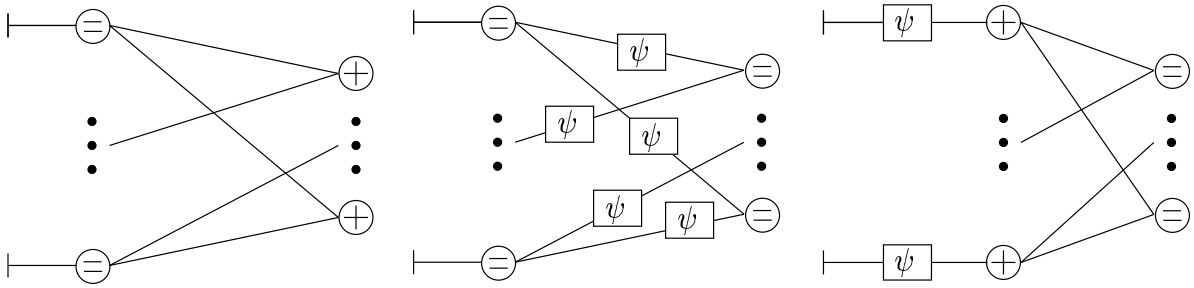
Similarly, the sum-product algorithm for arbitrary channels can be also performed equivalently on the dual LDPC graph provided that states are initialized to $(1, 0)$, initial messages (p_0, p_1) are Fourier-transformed to (p_0+p_1, p_0-p_1), and the results are inverse-Fourier-transformed [19]. We can also define dualized density evolution on the dual LDPC graph accordingly. However, we note that there is a big difference between dualized density evolution and the dualized reciprocal-channel approximation; namely the two message domains are different for the former but the same for the latter.

Table 7.1 shows the accuracy of the reciprocal channel approximation for AWGN channels. It is the best of five approximation methods (erasure-channel, Gaussian,



(a) Original (b) Equivalent calculation

Figure 7-12: Calculation of the output m_u using reciprocal-channel mappings at a check node



(a) Original LDPC code (b) Equivalent calculation (c) Another equivalent calculation

Figure 7-13: Equivalent calculations under the reciprocal-channel approximation. (a) is the original LDPC code. Parity checks are converted to repetition codes in (b) using reciprocal-channel mappings between the two sides. The dual code of the original LDPC code is used in (c).

Gaussian-capacity, pairwise Gaussian-capacity, and reciprocal-channel), and about ten times better than the Gaussian approximation. Figure 7-1 also shows the accuracy of the reciprocal-channel approximation for AWGN channels for various regular codes.

Unfortunately, we do not have a good reciprocal-channel approximation for the Laplace channel and for the the BSC. It does not seem that there is a natural one-dimensional parameter that represents the message density faithfully for these cases.

However, we can find the AWGN threshold of a code using the reciprocal-channel approximation and then map the threshold to λ (Laplace) or p (BSC) using equal-capacity curves. Tables 7.2 and 7.3 show the accuracy of this reciprocal-channel approximation for the Laplace and BSCs. We observe that the accuracy is comparable to that of the other two approximation methods, but not significantly better as in the AWGN case.

7.4.1 Gaussian and Reciprocal-Channel Approximations

It turns out that the Gaussian approximation and the reciprocal-channel approximation are very similar, although they were developed with different motivations. Figure 7-14 shows $\alpha = \frac{1}{\psi(x)} \log(1 - \phi(x))$ as a function of x, which is close to -0.27 for a wide range of x of our interest. Figure 7-15 shows plots of $\phi(x)$ and $1 - e^{\alpha\psi(x)}$, assuming $\alpha = -0.27$.

Although it seems that there is no natural way of generalizing the reciprocal-channel approximation to irregular codes, we can try replacing $\phi(x)$ with $1 - e^{\alpha\psi(x)}$ in (7.7) using the observation in Figure 7-14. This improves the accuracy of the Gaussian approximation slightly, but not greatly.

7.5 Summary

In this chapter, we have developed some approximation methods for analyzing density evolution for the sum-product algorithm. The erasure-channel approximation and optimization methods are based on approximating the target channel as the BEC. The Gaussian approximation method is motivated from empirical observations that

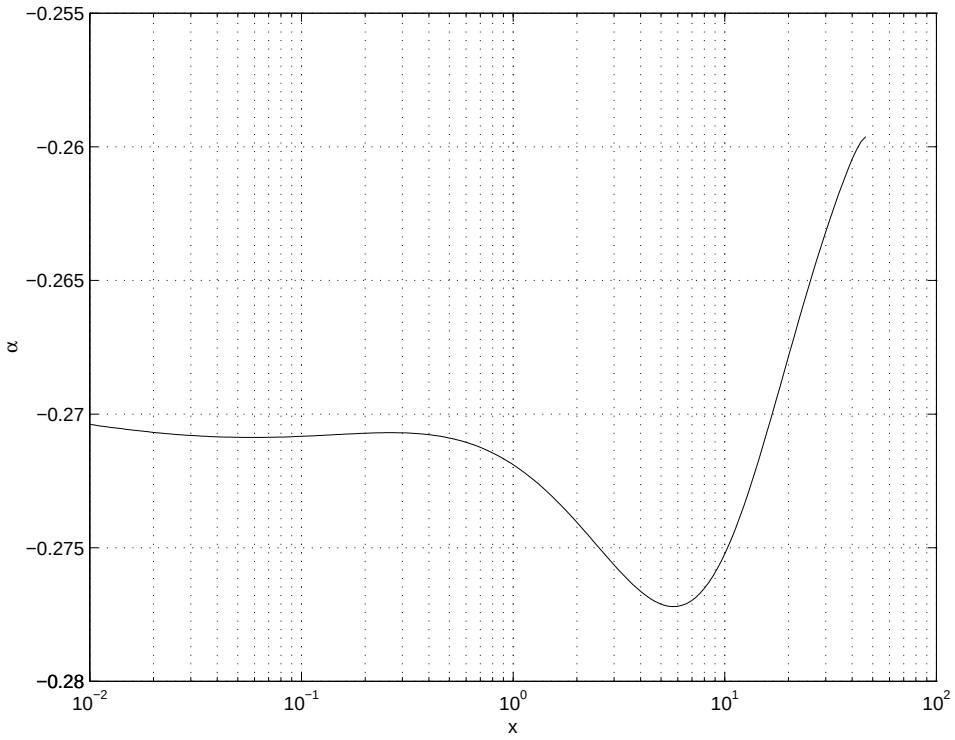


Figure 7-14: Plot of $\alpha = \frac{1}{\psi(x)} \log(1 - \phi(x))$

message distributions are Gaussian-like. Some analyses of density evolution based on the approximation have been shown. The Gaussian-capacity approximation is motivated by the observation in Chapter 4. We have also developed a very accurate model of the density evolution for AWGN channels, which is called the reciprocal-channel approximation.

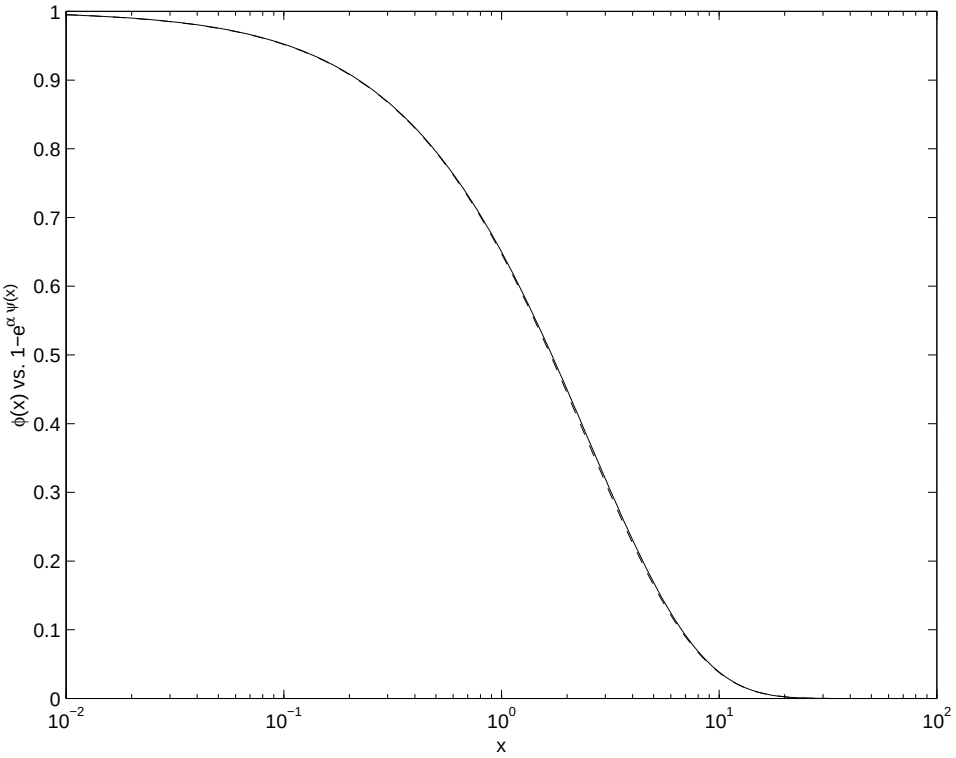


Figure 7-15: $\phi(x)$ (—) vs. $1 - e^{\alpha\psi(x)}$ (- -) using $\alpha = -0.27$. Note that two curves are very close.

Chapter 8

LDPC Code Design

8.1 Power-Limited Case

As observed in [53], degree distributions can be approximately optimized by running linear programming iteratively. In this thesis, we use a slightly different optimization technique to optimize $\lambda(x)$ (assuming $\rho(x)$ is given), which seems to be better for designing codes, especially when d_l is large. We maximize the rate of the code while maintaining some constraints:

1. $\lambda(1) = 1$ and $\lambda_i \geq 0$, $2 \leq i \leq d_l$;

2. the new $\lambda(x)$ is not significantly different from the old one (required to guarantee that the linear programming formulation is valid);

3. the new $\lambda(x)$ is better than the old one (produces smaller probability of error).

Let $\lambda(x) = \sum_{i=2}^{d_l} \lambda_i x^{i-1}$ denote the current left degree distribution and let $\lambda'(x) = \sum_{i=2}^{d_l} \lambda'_i x^{i-1}$ denote the updated (hopefully improved) left degree distribution.

Initially, we choose $\lambda(x)$ such that it has many high degrees, such as x^{d_l-1}, to make the initial rate low.

Let e_ℓ denote the probability of error (probability of being negative) of the input message to check nodes at the ℓ-th iteration when $\lambda(x)$ is used, i.e.,

$$e_\ell = \sum_{i=2}^{d_l} e_{\ell,i}\lambda_i,$$

where $e_{\ell,i}$ is the probability of error of the output message of the degree-i variable node at the ℓ-th iteration. Similarly, we define e'_ℓ as the probability of error of the input message to check nodes at the ℓ-th iteration when $\lambda(x)$ is used up to the $(\ell-1)$-st iteration and $\lambda'(x)$ is used at the ℓ-th iteration. Note that e'_ℓ is linear in λ'_i's, $i = 2, \ldots, d_l$.

We use the following for the constraint 2 above:

$$|e'_\ell - e_\ell| \le \max[0, \delta(e_{\ell-1} - e_\ell)],\ 1 \le \ell \le L,$$

where $\delta \ll 1$ is a small positive number and L is the maximum number of iterations. Note that this constraint is linear in λ'_i's. The "max" operation is not needed if the density evolution for the sum-product algorithm is perfect. However, since the probability of error can increase for the discretized density evolution for both the sum-product and the max-product algorithms, we need to make sure that the right hand side of the above inequality is nonnegative.

The following is the constraint 3 above:

$$e'_\ell \le e_{\ell-1},\ 1 \le \ell \le L.$$

Note that all constraints here and the objective function (rate) are linear in λ'_i's, $2 \le i \le d_l$.

We first fix $\rho(x)$ (we explain how to choose $\rho(x)$ later), then choose the initial $\lambda(x)$. We run the above linear programming several times until it converges (if it converges). If the resulting rate is not equal to the desired rate, then we change $\rho(x)$ and repeat these steps. Sometimes, we need to try several different initial $\lambda(x)$'s for better results.

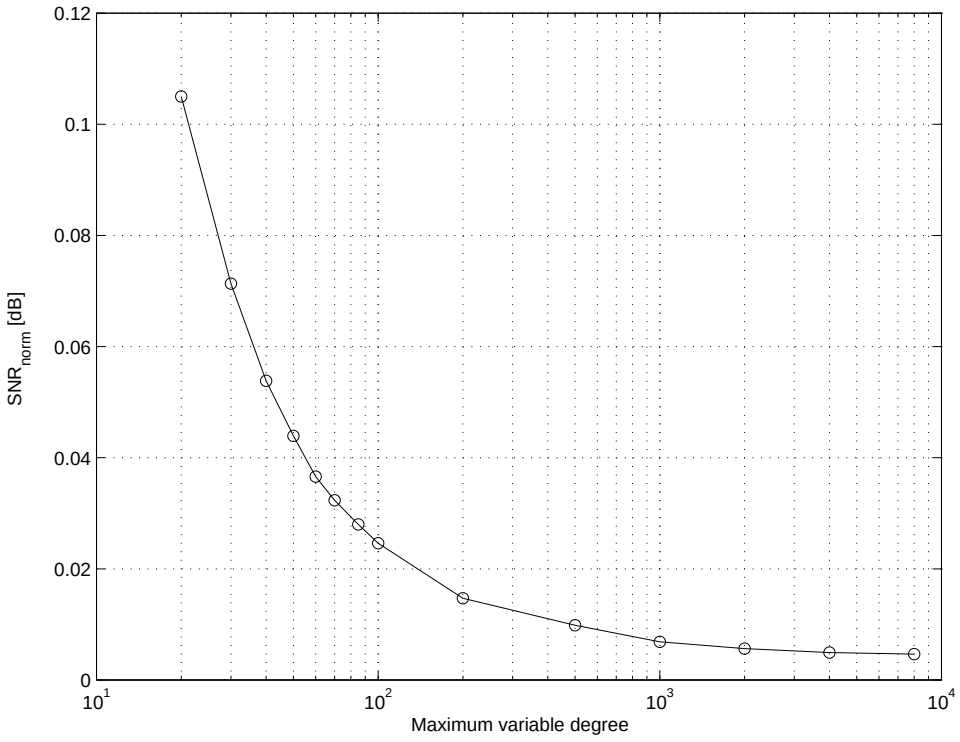


Figure 8-1: Threshold ($\mathrm{SNR}_{\mathrm{norm}}$) of rate-1/2 LDPC codes with maximum variable degree $= 20, \ldots, 8000$

Thresholds for some good rate-1/2 codes with maximum variable degrees 20, 30, 40, 50, 60, 70, 85, 100, 200, 500, 1000, 2000, 4000, and 8000 are given in Figure 8-1, where $\mathrm{SNR}_{\mathrm{norm}}$ is defined as the distance from the Shannon limit in dB. Tables 8.1, 8.2, 8.3, and 8.4 give the degree distributions for these codes, where threshold values are rounded down and $\mathrm{SNR}_{\mathrm{norm}}$ values are rounded up. The Shannon limit is $\sigma = 0.97869$ at rate 1/2. For $d_l = 500$ or higher, some restrictions on left degrees were placed for numerical stability. 12-bit quantization was used for the threshold calculation. Actual threshold values should be slightly higher.

We used concentrated $\rho(x)$'s only, where $\rho(x) = (1-\rho)x^{j-1} + \rho x^j$ for some integer $j \geq 2$. This restriction not only makes it easier to optimize $\rho(x)$, especially for large maximum variable degrees, but also is not too restrictive since codes optimized in this way are almost as good as codes with both degree distributions optimized. The average check degree ρ_{av} is used in Tables 8.1, 8.2, 8.3, and 8.4 to parametrize $\rho(x)$ where $\rho_{\mathrm{av}} = (1-\rho)(j-1) + \rho j = j - 1 + \rho$.

Table 8.1: Good rate-1/2 codes with $d_l = 20, 30, 40, 50$

d_l	20		30		40		50	
	x	λ_x	x	λ_x	x	λ_x	x	λ_x
	2	0.234029	2	0.209626	2	0.200642	2	0.191725
	3	0.212425	3	0.212527	3	0.189227	3	0.179895
	6	0.146898	6	0.013831	6	0.106155	6	0.121043
	7	0.102840	7	0.079586	7	0.091511	7	0.076094
	20	0.303808	8	0.195912	13	0.157193	14	0.016866
			30	0.288518	40	0.255272	15	0.167197
							50	0.247180
ρ_{av}	8.28125		8.95630		9.43140		9.81250	
σ	0.96693		0.97068		0.97265		0.97375	
SNR_{norm}	0.1051		0.0714		0.0539		0.0440	

Table 8.2: Good rate-1/2 codes with $d_l = 60, 70, 85, 100$

d_l	60		70		85		100	
	x	λ_x	x	λ_x	x	λ_x	x	λ_x
	2	0.185577	2	0.180410	2	0.173958	2	0.170031
	3	0.176146	3	0.171290	3	0.168303	3	0.160460
	6	0.068523	6	0.106619	6	0.081252	6	0.112837
	7	0.142803	7	0.053388	7	0.045809	7	0.047489
	17	0.116878	9	0.075460	8	0.072269	10	0.011481
	18	0.071893	20	0.132785	9	0.034193	11	0.091537
	60	0.238180	21	0.054359	22	0.109782	26	0.152978
			70	0.225689	23	0.095077	27	0.036131
					85	0.219357	100	0.217056
ρ_{av}	10.1010		10.3556		10.6763		10.9375	
σ	0.97457		0.97505		0.97554		0.97592	
SNR_{norm}	0.0366		0.0324		0.0280		0.0247	

Table 8.3: Good rate-1/2 codes with $d_l = 200, 500, 1000$

d_l	200		500		1000	
	x	λ_x	x	λ_x	x	λ_x
	2	0.153425	2	0.140509	2	0.124863
	3	0.147526	3	0.136314	3	0.121986
	6	0.041539	6	0.035248	6	0.039029
	7	0.147551	7	0.141107	7	0.119282
	18	0.047938	15	0.027721	15	0.032345
	19	0.119555	20	0.130839	20	0.115111
	55	0.036379	40	0.112991	50	0.093613
	56	0.126714	100	0.045886	70	0.042103
	200	0.179373	120	0.064591	150	0.100864
			320	0.128402	250	0.027667
			500	0.036392	450	0.085652
					1000	0.097485
ρ_{av}	12.0000		13.0000		14.5000	
σ	0.97704		0.97758		0.97792	
SNR_{norm}	0.0147		0.0099		0.0069	

In Figure 8-2, we show simulation results for the $d_l = 100, 200$ codes in Tables 8.2 and 8.3. A block length of 10^7 was used, and the code graph was constructed randomly, except that cycles of length 2 and 4 were avoided. At BER = 10^{-6}, the $d_l = 200$ code operates within 0.04 dB of the Shannon limit.

The maximum allowed number of iterations was 2000. When decoding was successful, about 800–1100 iterations were needed. No undetected errors occurred.

At a BER of about 10^{-6}, we decoded 249 and 371 blocks for the $d_l = 100$ and $d_l = 200$ codes, respectively— i.e., about 1.2×10^9 and 1.8×10^9 decoded bits, respectively.

Unfortunately, because of the large block length, we have been unable to run more simulations to reduce the uncertainty in the BER. However, since a slight increase in the SNR, say 0.005 dB, reduces the BER dramatically, we believe these simulation results predict the actual performance of each code reasonably well.

Table 8.4: Good rate-1/2 codes with $d_l = 2000, 4000, 8000$

d_l	2000		4000		8000	
	x	λ_x	x	λ_x	x	λ_x
	2	0.112355	2	0.105332	2	0.096294
	3	0.110498	3	0.103764	3	0.095393
	6	0.032724	6	0.035237	6	0.033599
	7	0.112381	7	0.100941	7	0.091918
	15	0.026831	15	0.029112	15	0.031642
	20	0.107729	20	0.099561	20	0.086563
	50	0.086616	50	0.084268	50	0.093896
	70	0.038843	70	0.034889	70	0.006035
	150	0.080155	150	0.088540	100	0.018375
	200	0.037178	250	0.027687	150	0.086919
	500	0.109602	400	0.059494	400	0.089018
	900	0.011150	900	0.097641	900	0.057176
	2000	0.133938	3000	0.070928	2000	0.085816
			4000	0.062606	3000	0.006163
					6000	0.003028
					8000	0.118165
ρ_{av}	16.0000		17.0000		18.5000	
σ	0.97805		0.978137		0.9781869	
SNR_{norm}	0.0057		0.00495		0.00450	

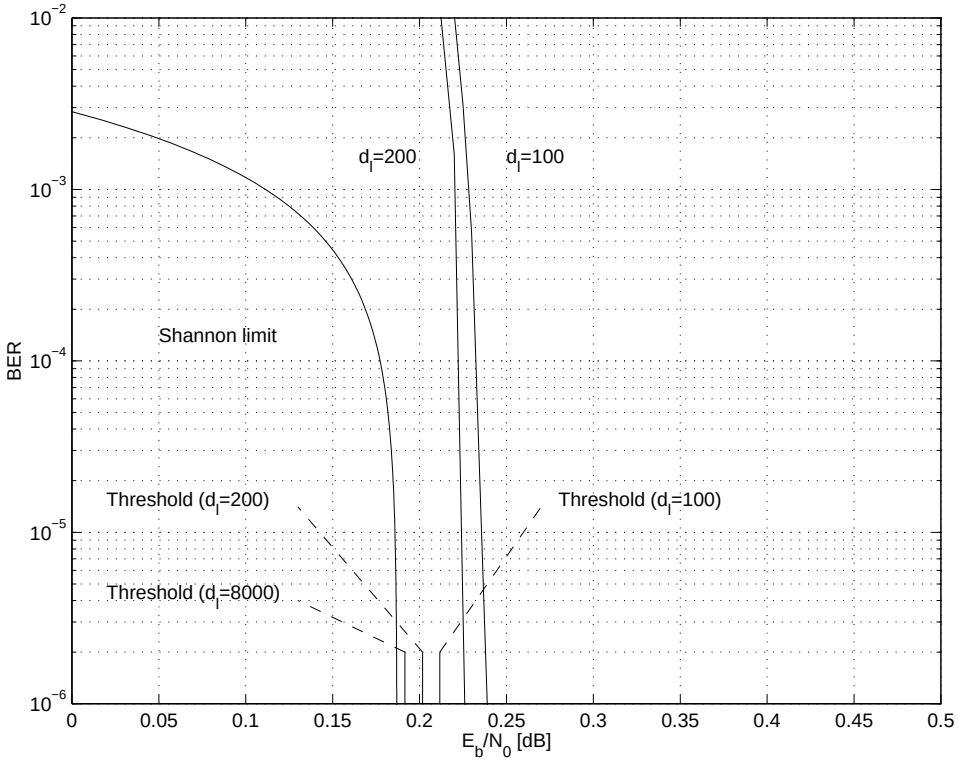


Figure 8-2: Simulation results for the $d_l = 100$ and $d_l = 200$ codes of Tables 8.2 and 8.3, using a block length of 10^7.

8.2 Bandwidth-Limited Case

Wachsmann and Huber [68] used the idea of multilevel coding [37] to achieve a very good coding scheme by using turbo codes, as component codes as shown in Figure 8-3. They proved that 2^m-ary multilevel coding with multistage decoding can be equivalently decomposed into m binary-input channels, in which the channel capacity of the multilevel channel is equal to the sum of the channel capacities of the component binary-input channels. This implies that the channel capacity can be asymptotically achieved via multilevel coding with multistage decoding using capacity-achieving binary codes at each level.

In [68], punctured turbo codes were used to achieve the desired rates for each level. For high rates close to 1, a single-parity-check code (especially for small block lengths) was used without noticeable performance degradation. It was shown that Ungerboeck's set partitioning produced the best results. However, it was also shown that different set-partitioning schemes are not much different in performance, which is due to the fact that the capacity-achieving theorem holds for any set partitioning.

Multistage decoding was performed from the top level (nosiest) to the bottom level by using decisions of higher levels. To minimize error propagation due to re-encoding of the data especially for low-rate turbo codes, symbol-by-symbol MAP was used to estimate parity bits directly.

The biggest advantage of this scheme is that we can use binary codes to design coding schemes for bandwidth-limited channels and obtain flexibility in the rate, spectral efficiency, and modulation type. This, however, in turn requires high encoding and decoding complexity.

In Figure 8-4, a 2-level channel example from [22] is shown, where we assume that inputs X_1 and X_2 are both equiprobable. This can be easily generalized to m levels. We consider 2 types of channels, one having Y as an output and the other U as an output. We assume that Z is an AWGN that is independent of inputs.

Using the chain rule of information theory, we can decompose a multilevel channel into an equivalent set of smaller channels. When Y is the channel output, 2 types of

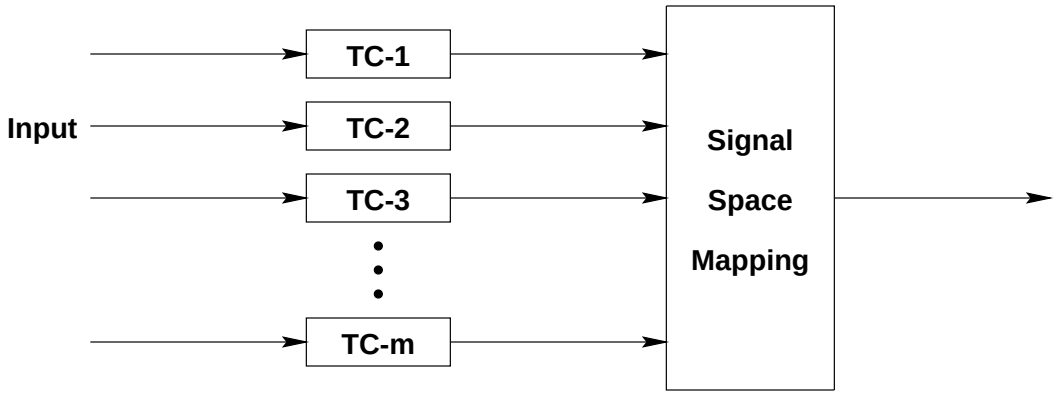


Figure 8-3: Multilevel turbo coded modulation

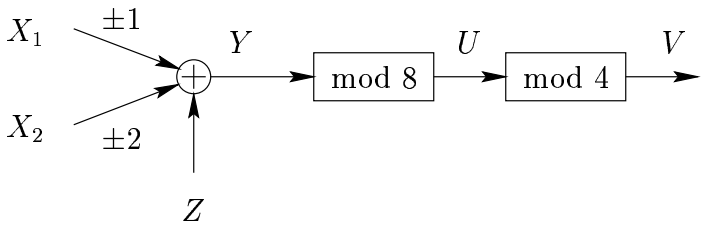


Figure 8-4: 2-level channel example

decompositions are possible, as observed in [22]:

$$I(X_1X_2;Y) = I(X_1;Y) + I(X_2;Y|X_1)$$

and

$$I(X_1X_2;Y) = I(X_2;Y) + I(X_1;Y|X_2)$$

The decoder can first decode $X_1(X_2)$, regarding $X_2(X_1)$ as noise, respectively. The remaining information $X_2(X_1)$ can then be retrieved given $X_1(X_2)$, respectively, if the first decoding stage is successful.

When U is the channel output, we use the following decomposition:

$$I(X_1X_2;U) = I(X_1;U) + I(X_2;U|X_1).$$

Decoding based on "stripping" as in the previous example can be used to utilize this

decomposition. It is interesting to see that $I(X_1;U) = I(X_1;V)$, which means that X_1 is not affected by X_2, this makes this decomposition strongly preferable [22] to the other type of decomposition in which X_2 is decoded first.

In this thesis, we consider both types of channels, with decoding order from the top to the bottom levels. We can use LDPC codes at each level of the multilevel coding scheme. In each case, we note that the equivalent channel for each level is symmetric, i.e., $p(y|x_1 = 1) = p(-y|x_1 = -1)$, $p(u|x_1 = 1) = p(-u|x_1 = -1)$, $p(y|x_2 = 2, x_1 = 0) = p(-y|x_2 = -2, x_1 = 0)$, and $p(u|x_2 = 2, x_1 = 0) = p(-u|x_2 = -2, x_1 = 0)$, where w.l.o.g. we assume $x_1 = 0$ for the bottom level since x_1 is known. Therefore, we can use the same technique as in the last section to optimize degree distributions. We show that by optimizing LDPC codes for each level, capacity-approaching performance is obtained.

Although, we can design LDPC codes for all levels, we note that powerful codes are needed only at the top level, and simple algebraic codes should be enough at lower levels where the rates are close to 1, as observed in [22].

Although, the initial densities for the two types of channels and for each level are very different, we show that optimized distributions are interchangeable without much degradation in performance.

Table 8.5 shows some good rate-1/2 degree distributions with $d_l = 20$. The code in the first column is designed for the ordinary AWGN channel (from Table 8.1). The codes in the second and the third columns are designed for the channels with input X_1 and output V and Y in Figure 8-4, respectively.

When threshold values are evaluated not only for the target channel but also for the other two channels, they are very similar. This verifies the observations in Section 6.3 on the predictability of thresholds for other channels. However, by optimizing the code for the target channel, we get slightly better results.

It seems codes designed for the ordinary AWGN and the noisiest level of 4-PAM channel are interchangeable without much difference. Motivated from this, we used the $d_l = 4000$ code in Table 8.4 to calculate thresholds for each channel, which is the last column in Table 8.5. In fact, it shows very good performance for the PAM

Table 8.5: Good rate-1/2 codes with $d_l = 20$ designed for AWGN, mod-AWGN, the highest level of 4-PAM, and AWGN ($d_l = 4000$), from the second to the fifth columns, respectively. Threshold values for AWGN, mod-AWGN, and PAM channels are given in terms of $\mathrm{SNR}_{\mathrm{norm}}$ in dB. All SNR values are rounded up.

Target	AWGN		mod-AWGN		PAM		AWGN
	x	λ_x	x	λ_x	x	λ_x	
	2	0.234029	2	0.218744	2	0.231404	
	3	0.212425	3	0.205996	3	0.205101	$d_l = 4000$
	6	0.146898	6	0.183885	6	0.201006	code in
	7	0.102840	7	0.056575	7	0.040011	Table 8.4
	20	0.303808	20	0.334800	20	0.322478	
ρ_{av}	8.28125		8.59375		8.38125		17.0000
AWGN	0.106		0.173		0.110		0.0050
mod-AWGN	0.166		0.071		0.176		0.114
PAM	0.079		0.127		0.079		0.0063

channel, i.e., it is only 0.0063 dB from the channel capacity of the noisiest level of 4-PAM.

8.3 Summary

In this chapter, we have designed good LDPC codes for AWGN channels. For binary-input AWGN channels (power-limited), our best code has a threshold within 0.0045 dB of the Shannon limit. Simulation results show that we can achieve within 0.04 dB of the Shannon limit at a bit error rate of 10^{-6} using a block length of 10^7.

We have also designed some good LDPC codes for bandwidth-limited channels that are very close to the Shannon limit.

Chapter 9

Conclusions

We have constructed various coding schemes that approach the Shannon limit very closely.

Several different types of source and channel pairs were introduced and analyzed. Some of the sources and channels were matched, which means that an uncoded scheme can achieve the Shannon limit for all values of SNR. Since there is no parameter to be adjusted in the uncoded system when the channel parameter changes, such a system is universal in some sense. We have constructed the rate-distortion function of a uniform source over a fundamental region of a lattice with a modulo-difference-distortion measure. We have also shown the optimality of an uncoded system for the uniform source and a modulo white additive noise channel.

When the source and channel are not matched, especially when the source bandwidth is greater than the channel bandwidth, we have used space-filling curves and other families of curves to construct analog joint source-channel coding schemes that operate very close to the Shannon limit. For uniform analog sources with 2:1 bandwidth reduction, we were able to achieve within 1.1 dB of the rate-distortion limit for a wide range of SNR using several space-filling curves. For Gaussian sources with 2:1 bandwidth reduction, we have shown that it is possible to achieve within 0.9 dB of the rate-distortion limit using spiral-like curves.

Since there is no delay in these analog coding schemes, they can be used in an environment where the noise characteristics change very rapidly. Although we did

not construct any practical systems using our schemes, these schemes may be used for compressing video and audio signals and for transmitting them through an analog channel. Since such source signals have high redundancy, applying transform coding would be desirable before using our technique.

For LDPC codes and sum-product decoding, we have constructed some approximation methods to analyze density evolution. Although they were developed with different motivations, they are all reasonably accurate. Using these approximation methods, we have simplified density evolution to one-dimensional problems, and found simple and analyzable expressions for describing the approximate density evolution. Because of the huge reduction in the dimension of the problem without much sacrifice in accuracy, we were able to find thresholds faster and optimize degree distributions faster and easier. These approximations (especially the Gaussian approximation) are also good tools to visualize how density evolution behaves, and to get some hints as to how to optimize degree distributions. Graphically, we have demonstrated how the probability of error approaches zero.

We have shown how density evolution works for the max-product algorithm. There are many interesting aspects of density evolution with the max-product algorithm.

We have designed LDPC codes for binary-input AWGN channels that approach the Shannon limit very closely. Our design was based on discretized density evolution and optimization of degree distributions via iterative linear programming. Our methods can be used to design LDPC codes of arbitrary rates between 0 and 1 for many interesting channels, including the BEC, the BSC, and the Laplace and AWGN channels.

For rate 1/2, the best code found has a threshold within 0.0045 dB of the Shannon limit of the binary-input AWGN channel. Simulation results show that we can achieve within 0.04 dB of the Shannon limit at a bit error rate of 10^{-6} using a block length of 10^7.

Our results strongly suggest that optimal LDPC codes for AWGN channels may approach the Shannon limit asymptotically as the block length tends to infinity.

We have also designed rate 1/2 LDPC codes for bandwidth-limited channels that

have thresholds within 0.0063 dB of the Shannon limit of the nosiest level of the 4-PAM AWGN channel.

9.1 Thesis Contributions

- Matched sources and channels have been characterized.
- The rate-distortion function of a uniform source on a fundamental region of a lattice with a modulo-difference-distortion measure has been characterized.
- Capacity-achieving joint source-channel coding schemes have been constructed for uniform sources and modulo channels.
- Capacity-approaching joint source-channel coding schemes have been constructed and analyzed when the source bandwidth is greater than the channel bandwidth, using space-filling and other families of curves.
- Normalization of symmetric channels.
- Density evolution for the max-product algorithm has been characterized.
- Erasure-channel, Gaussian-capacity, and reciprocal-channel approximations have been constructed, based on the observation that threshold values can be accurately mapped between channels.
- A Gaussian approximation has been developed for the analysis of the sum-product algorithm for LDPC codes.
- A rough measure of easiness of designing LDPC codes for binary-input symmetric channels has been proposed and characterized.
- A fast and accurate discretized density evolution algorithm has been proposed.
- Good LDPC codes that approach the Shannon limit within 0.0045 dB asymptotically have been constructed. Good LDPC codes that approach the Shannon limit within 0.04 dB have been demonstrated by simulations.

- Good LDPC codes for two multilevel coding schemes have been constructed.

9.2 Future Work

We propose the following future research:

- More examples of capacity-achieving joint source-channel coding schemes.
- Joint source-channel coding schemes using space-filling curves with higher bandwidth-reduction ratio.
- Joint source-channel coding schemes using some other types of curves.
- Generalization of the joint source-channel coding results to a broader class of sources and channels.
- An exact one-dimensional representation of density evolution at least for some channels.
- A capacity-achieving sequence of degree distributions for the AWGN channel.
- More analysis on why the reciprocal-channel approximation works so well.
- Design of capacity-approaching LDPC codes for short block lengths.

Appendix A

Proofs in Chapter 2

In this appendix, we provide proofs of Lemmas 2.4 and 2.5 and Theorem 2.6. We restate each lemma and theorem here for reader's convenience.

Lemma A.1 (2.4) *The differential entropy $h(f_{d_\Lambda,D})$ of a truncated exponential density function $f_{d_\Lambda,D}$ increases monotonically as D increases if $\mu > 0$, and decreases monotonically as D increases if $\mu < 0$. Furthermore, it is uniquely determined for a given D.*

Proof: Assume that there are two such functions, namely f_{d_Λ,D_1} and f_{d_Λ,D_2}. Let $\{c_1, \mu_1\}$ and $\{c_2, \mu_2\}$ denote the parameters of f_{d_Λ,D_1} and f_{d_Λ,D_2}, respectively. Using the fact that the Kullback-Leibler distance $D(\cdot\|\cdot)$ is always nonnegative, we get

$$\begin{aligned}
0 &\leq D(f_{d_\Lambda,D_1}\|f_{d_\Lambda,D_2}) \\
&= \int_{\mathcal{R}(\Lambda)} f_{d_\Lambda,D_1}(\mathbf{x}) \log \frac{f_{d_\Lambda,D_1}(\mathbf{x})}{f_{d_\Lambda,D_2}(\mathbf{x})}\, d\mathbf{x} \\
&= -h(f_{d_\Lambda,D_1}) - \int_{\mathcal{R}(\Lambda)} f_{d_\Lambda,D_1}(\mathbf{x}) \log f_{d_\Lambda,D_2}(\mathbf{x})\, d\mathbf{x} \\
&= -h(f_{d_\Lambda,D_1}) - \int_{\mathcal{R}(\Lambda)} f_{d_\Lambda,D_1}(\mathbf{x}) \left[\log c_2 - \mu_2 d_\Lambda(\mathbf{x})\right]\, d\mathbf{x} \\
&= -h(f_{d_\Lambda,D_1}) - [\log c_2 - \mu_2 D_1] \\
&= -h(f_{d_\Lambda,D_1}) - [\log c_2 - \mu_2 D_2] + \mu_2[D_1 - D_2] \\
&= -h(f_{d_\Lambda,D_1}) + h(f_{d_\Lambda,D_2}) + \mu_2[D_1 - D_2].
\end{aligned}$$

Hence we get

$$h(f_{d_\Lambda,D_2}) \geq h(f_{d_\Lambda,D_1}) + \mu_2[D_2 - D_1].$$

Exchanging indices 1 and 2, we also get

$$h(f_{d_\Lambda,D_1}) \geq h(f_{d_\Lambda,D_2}) + \mu_1[D_1 - D_2].$$

Hence if $D_1 = D_2$, then $h(f_{d_\Lambda,D_1}) = h(f_{d_\Lambda,D_2})$. If $D_1 \neq D_2$, we first assume that $\mu_1 > 0$ and $\mu_2 > 0$; then we get

$$h(f_{d_\Lambda,D_1}) > h(f_{d_\Lambda,D_2}) \qquad \text{if } D_1 > D_2$$

using the second inequality, and we get

$$h(f_{d_\Lambda,D_1}) < h(f_{d_\Lambda,D_2}) \qquad \text{if } D_1 < D_2$$

using the first inequality. Therefore $h(f_{d_\Lambda,D})$ increases monotonically as D increases if $\mu > 0$. Similarly assume that $\mu_1 < 0$ and $\mu_2 < 0$; then we get

$$h(f_{d_\Lambda,D_2}) > h(f_{d_\Lambda,D_1}) \qquad \text{if } D_1 > D_2$$

using the first inequality, and we get

$$h(f_{d_\Lambda,D_2}) < h(f_{d_\Lambda,D_1}) \qquad \text{if } D_1 < D_2$$

using the second inequality. Therefore $h(f_{d_\Lambda,D})$ decreases monotonically as D increases if $\mu < 0$. □

Lemma A.2 (2.5) *Let the random vectors $\mathbf{X}, \mathbf{Y} \in \mathcal{R}(\Lambda) \subset \mathbb{R}^n$ have expected distortion D, i.e., $E[d_\Lambda(\mathbf{X})] = D$ and $E[d_\Lambda(\mathbf{Y})] = D$. Let $\mathbf{Y}$ have the truncated exponential density function associated with $d_\Lambda(\cdot)$. Then $h(\mathbf{X}) \leq h(\mathbf{Y})$, with equality iff $\mathbf{X} = \mathbf{Y}$ almost everywhere.*

Proof: Let g be the density function of $\mathbf{X}$ satisfying $\int_{\mathcal{R}(\Lambda)} d_\Lambda(\mathbf{x}) g(\mathbf{x})\, d\mathbf{x} = D$ and let $f_{d_\Lambda,D}(\mathbf{y})$ be the density function of $\mathbf{Y}$. Then

$$\begin{aligned} 0 &\leq D(g \| f_{d_\Lambda,D}) \\ &= \int_{\mathcal{R}(\Lambda)} g(\mathbf{x}) \log \frac{g(\mathbf{x})}{f_{d_\Lambda,D}(\mathbf{x})}\, d\mathbf{x} \\ &= -h(g) - \int_{\mathcal{R}(\Lambda)} g(\mathbf{x}) \log f_{d_\Lambda,D}(\mathbf{x})\, d\mathbf{x} \\ &= -h(g) - \int_{\mathcal{R}(\Lambda)} f_{d_\Lambda,D}(\mathbf{x}) \log f_{d_\Lambda,D}(\mathbf{x})\, d\mathbf{x} \\ &= -h(g) + h(f_{d_{\Lambda,D}}), \end{aligned}$$

where $\int_{\mathcal{R}(\Lambda)} g(\mathbf{x}) \log f_{d_\Lambda,D}(\mathbf{x})\, d\mathbf{x} = \int_{\mathcal{R}(\Lambda)} f_{d_\Lambda,D}(\mathbf{x}) \log f_{d_\Lambda,D}(\mathbf{x})\, d\mathbf{x}$ follows from the fact that g and $f_{d_{\Lambda,D}}$ have the same expected distortion. □

Theorem A.3 (2.6) *The rate-distortion function $R(D)$ of an iid uniform source $\mathbf{W}$ in $\mathcal{R}(\Lambda)$ with modulo difference distortion $d_\Lambda(\cdot,\cdot)$ is*

$$R(D) = \begin{cases} \log V(\Lambda) - h(f_{d_\Lambda,D}) & \text{if } D \leq D_0 \\ 0 & \text{otherwise.} \end{cases} \tag{A.1}$$

Proof: The rate-distortion function of $\mathbf{W}$ is

$$R(D) = \min_{p(\hat{\mathbf{w}}|\mathbf{w}):\ E[d_\Lambda(\mathbf{w},\hat{\mathbf{w}})] \leq D} I(\mathbf{W}; \hat{\mathbf{W}}).$$

First, we find a lower bound for the rate-distortion function and then show that it is

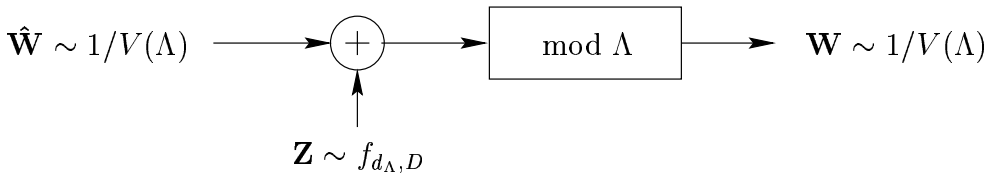


Figure A-1: Test channel.

achievable.

$$\begin{aligned}
I(\mathbf{W};\hat{\mathbf{W}}) &= h(\mathbf{W}) - h(\mathbf{W}|\hat{\mathbf{W}}) \\
&= \log V(\Lambda) - h((\mathbf{W}-\hat{\mathbf{W}}) \bmod \Lambda|\hat{\mathbf{W}}) \\
&\geq \log V(\Lambda) - h((\mathbf{W}-\hat{\mathbf{W}}) \bmod \Lambda) \\
&\geq \log V(\Lambda) - h(f_{d_\Lambda,E[d_\Lambda((\mathbf{W}-\hat{\mathbf{W}}) \bmod \Lambda)]}) \\
&= \log V(\Lambda) - h(f_{d_\Lambda,E[d_\Lambda(\mathbf{W},\hat{\mathbf{W}})]}) \\
&\geq \log V(\Lambda) - \begin{cases} h(f_{d_\Lambda,D}), & \text{if } D \leq D_0; \\ \log V(\Lambda), & \text{otherwise;} \end{cases} \\
&= \begin{cases} \log V(\Lambda) - h(f_{d_\Lambda,D}), & \text{if } D \leq D_0; \\ 0, & \text{otherwise,} \end{cases}
\end{aligned}$$

where the first inequality follows from the fact that conditioning reduces entropy, the second follows from Lemma 2.5, and the last follows from Lemma 2.4. Hence

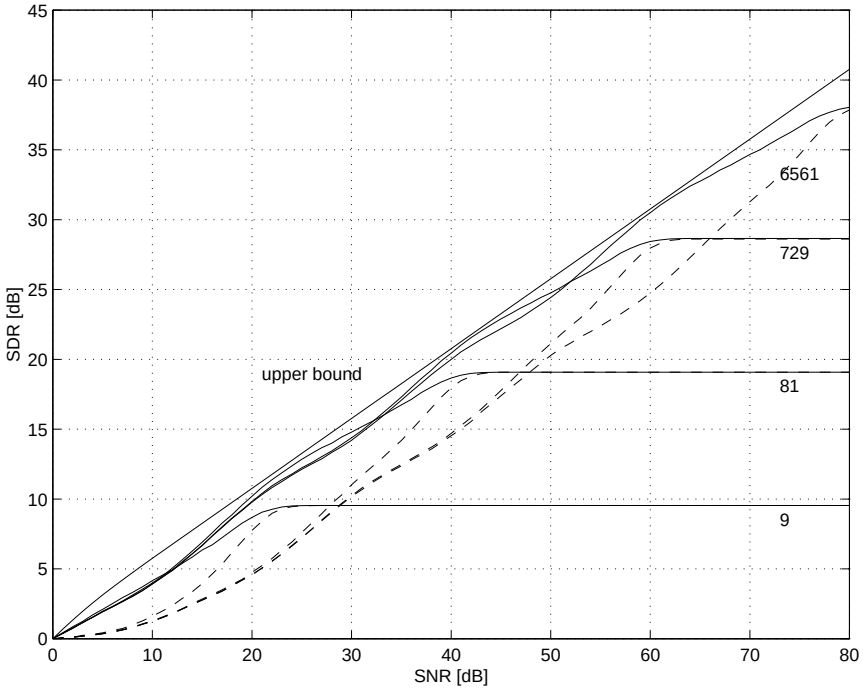
$$R(D) \geq \begin{cases} \log V(\Lambda) - h(f_{d_\Lambda,D}) & \text{if } D \leq D_0 \\ 0 & \text{otherwise.} \end{cases}$$

If $D \leq D_0$, then we can achieve this bound by constructing $\mathbf{W}$ from $\hat{\mathbf{W}}$ using an additive noise $\mathbf{Z}$ with pdf $f_{d_{\Lambda,D}}$ followed by a mod-Λ operation, where the noise $\mathbf{Z}$ and the input $\hat{\mathbf{W}}$ are independent and $\hat{\mathbf{W}}$ is uniform on $\mathcal{R}(\Lambda)$. If $D > D_0$, then we choose $\hat{\mathbf{W}} = 0$, thus achieving $R(D) = 0$. Figure A-1 represents the test channel to construct $\mathbf{W}$ from $\hat{\mathbf{W}}$, where $\hat{\mathbf{W}} \sim 1/V(\Lambda)$ means that $\hat{\mathbf{W}}$ has a uniform density $1/V(\Lambda)$ over $\mathcal{R}(\Lambda)$. $\square$

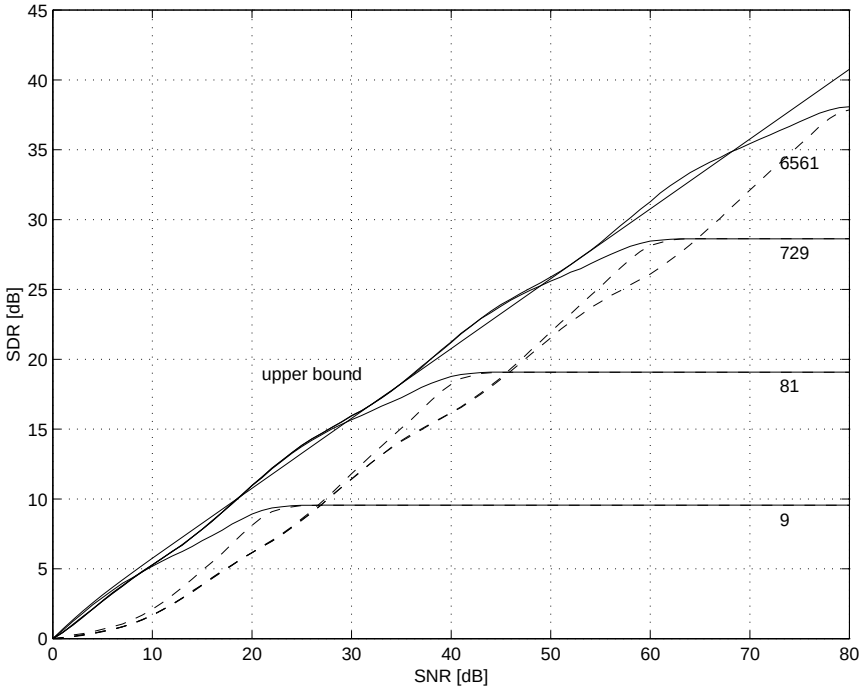
Appendix B

Simulation Results for Joint Coding Using Lattice Vector Quantizers

In this appendix, we show Monte Carlo simulation results for 2:1 and 3:1 bandwidth reduction for uniform sources using some space-filling curves. Curves used are the Peano curve in Figure 3-7, the 2D and 3D Hilbert curves in Figure 3-8, and the hexagonal space-filling curve in Figure 3-9. Lattice vector quantizers using approximate curves are used for quantization. Two types of decoders, namely the inverse-mapping and MMSE decoders, are used for each case. Figures B-1–B-6 show these simulation results.

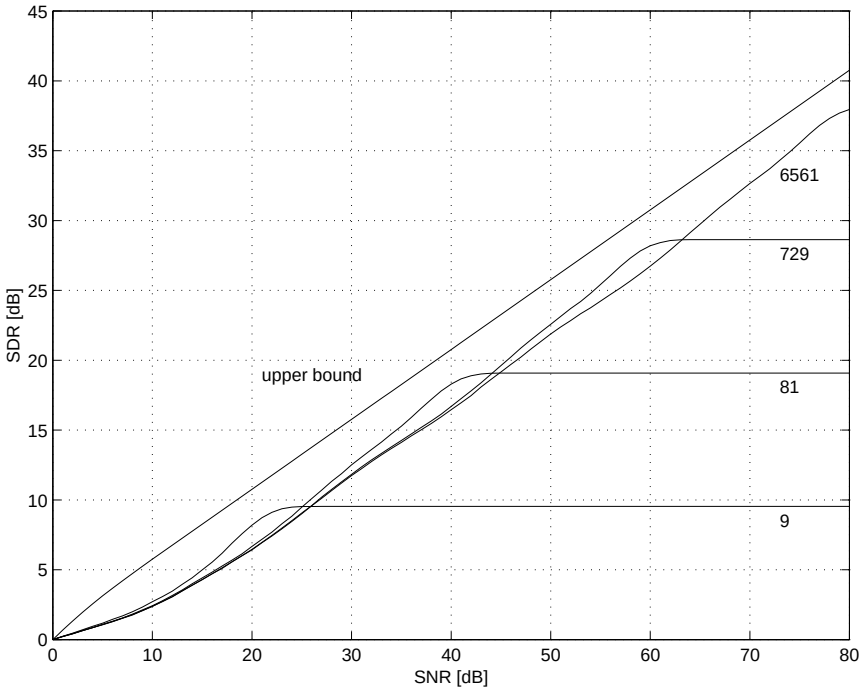


(a) Inverse-mapping decoder

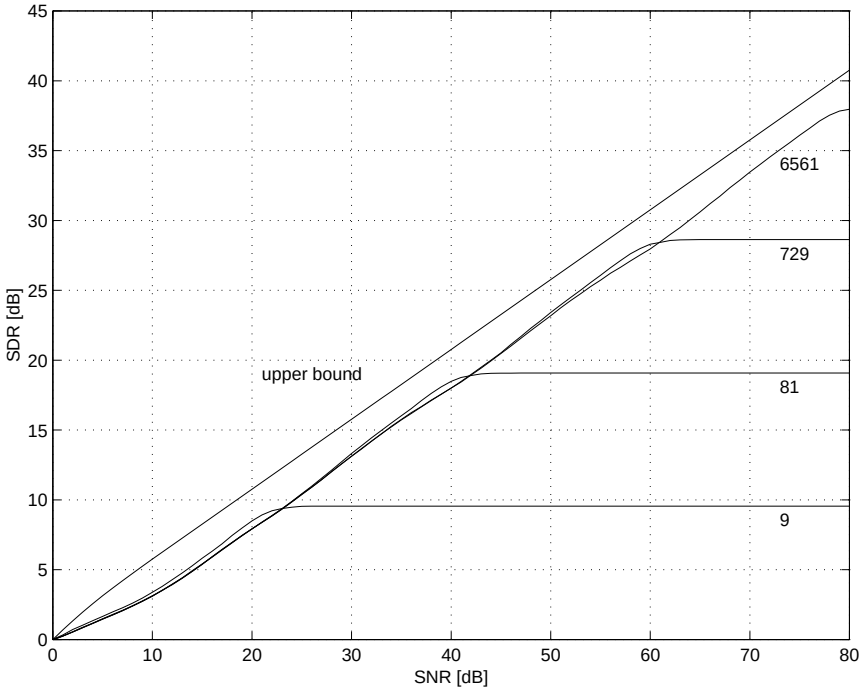


(b) MMSE decoder

Figure B-1: 2:1 bandwidth reduction using the Peano curve for uniform sources. A lattice vector quantizer is used for quantization (number of points of each case shown). SDRs for W_1 (—) and W_2 (- -) are plotted as functions of the channel SNR.

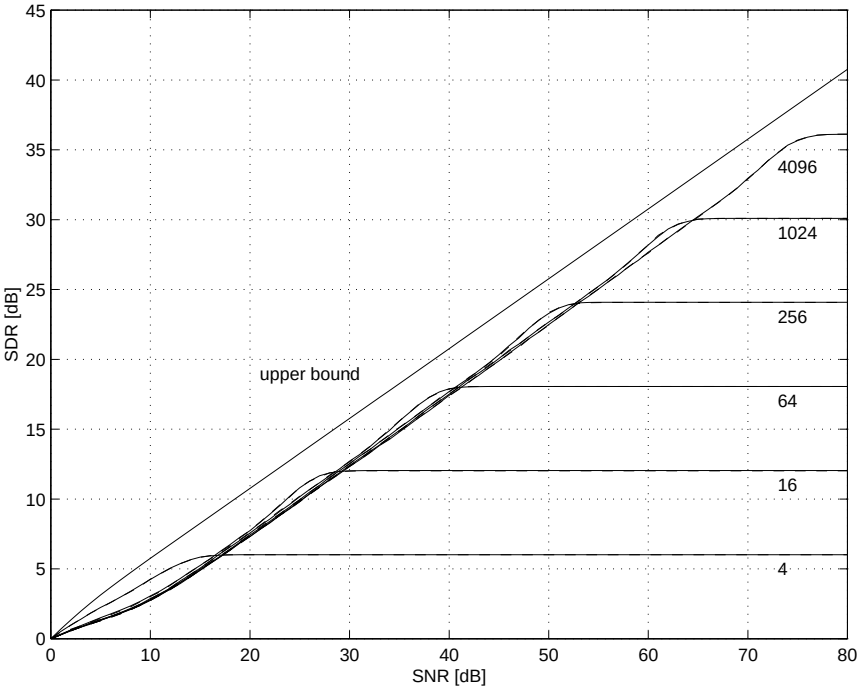


(a) Inverse-mapping decoder

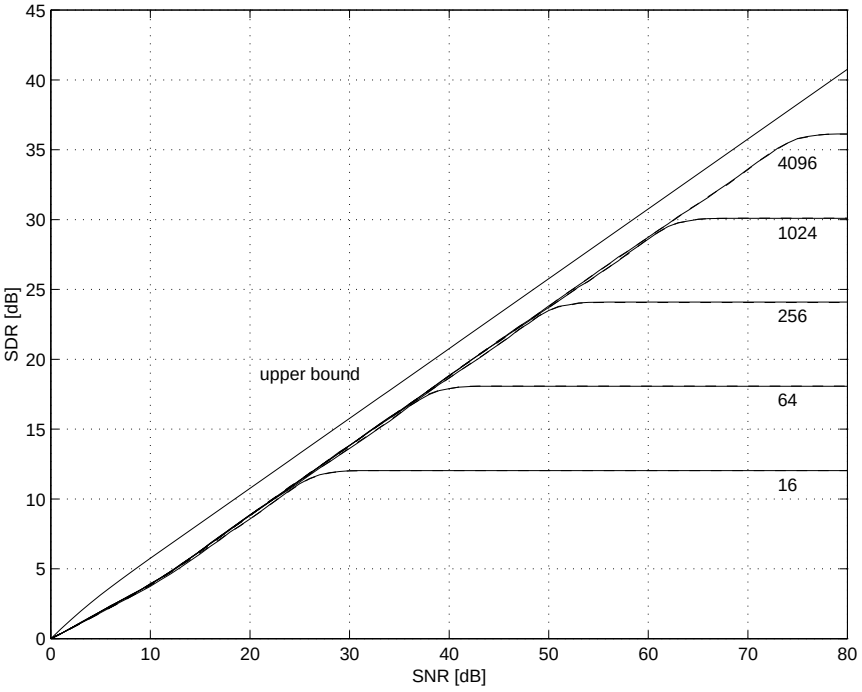


(b) MMSE decoder

Figure B-2: 2:1 bandwidth reduction using the Peano curve for uniform sources. A lattice vector quantizer is used for quantization (number of points of each case shown). Average SDRs are plotted as functions of the channel SNR.

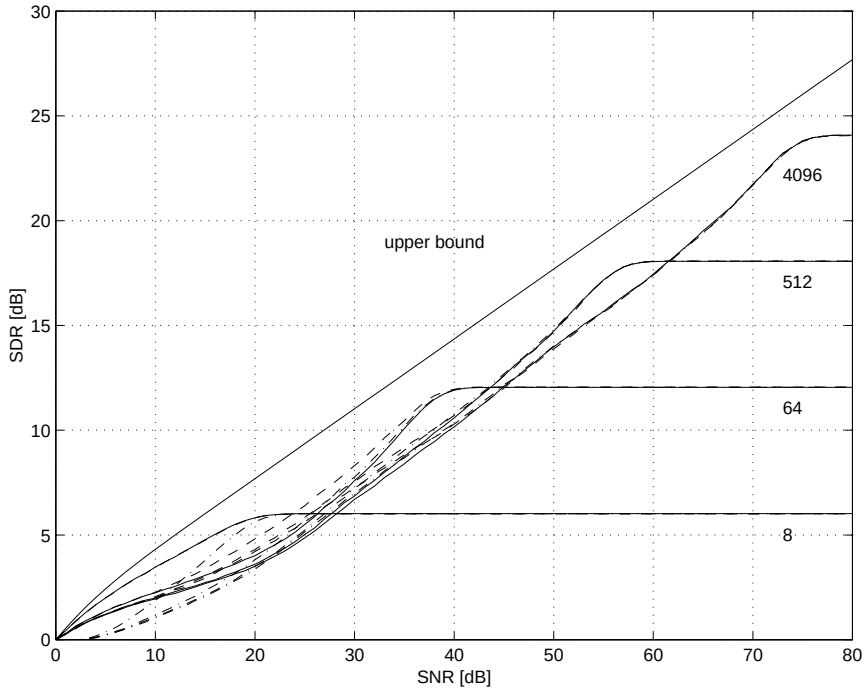


(a) Inverse-mapping decoder

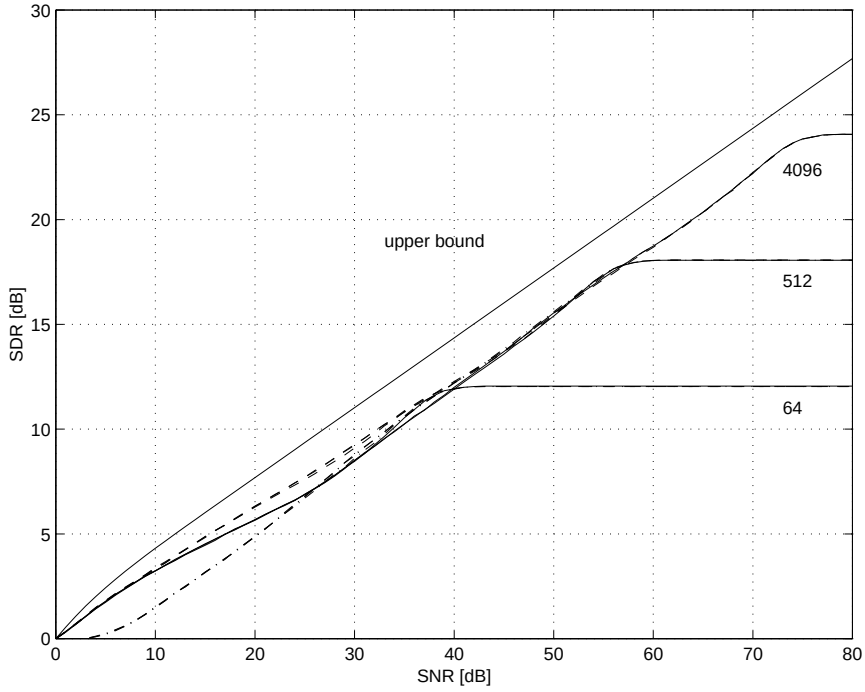


(b) MMSE decoder

Figure B-3: 2:1 bandwidth reduction using the Hilbert curve for uniform sources. A lattice vector quantizer is used for quantization (number of points of each case shown). SDRs for W_1 (—) and W_2 (- -) are plotted as functions of the channel SNR.

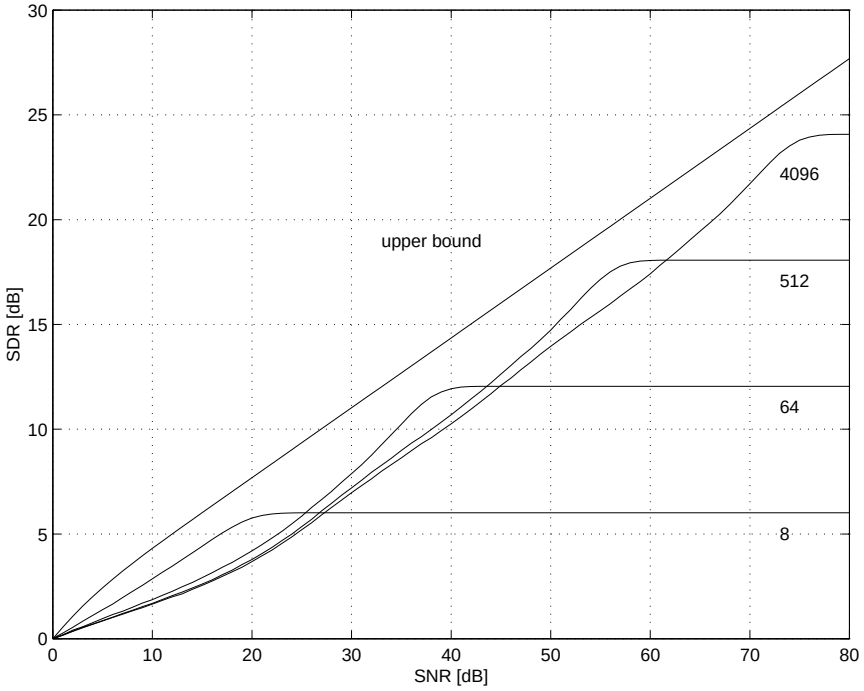


(a) Inverse-mapping decoder

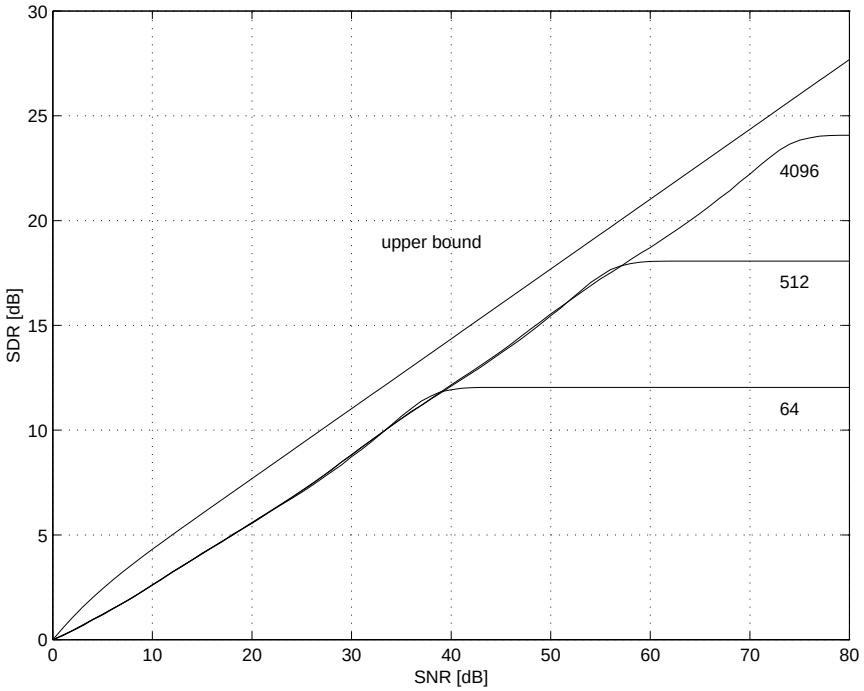


(b) MMSE decoder

Figure B-4: 3:1 bandwidth reduction using the three-dimensional Hilbert curve for uniform sources. A lattice vector quantizer is used for quantization (number of points of each case shown). SDRs for W_1 (—), W_2 (- -), and W_3 (- ·) are plotted as functions of the channel SNR.

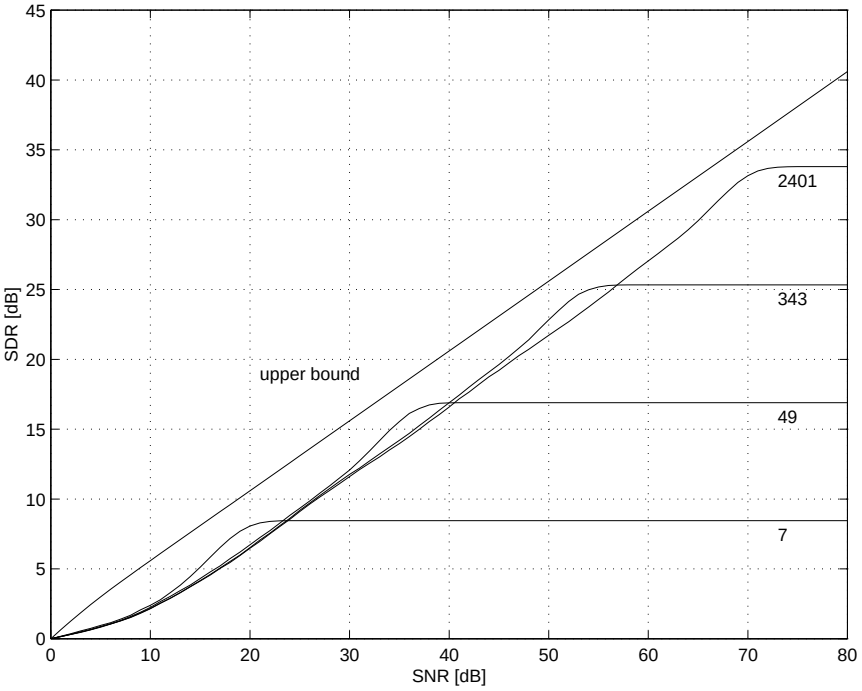


(a) Inverse-mapping decoder

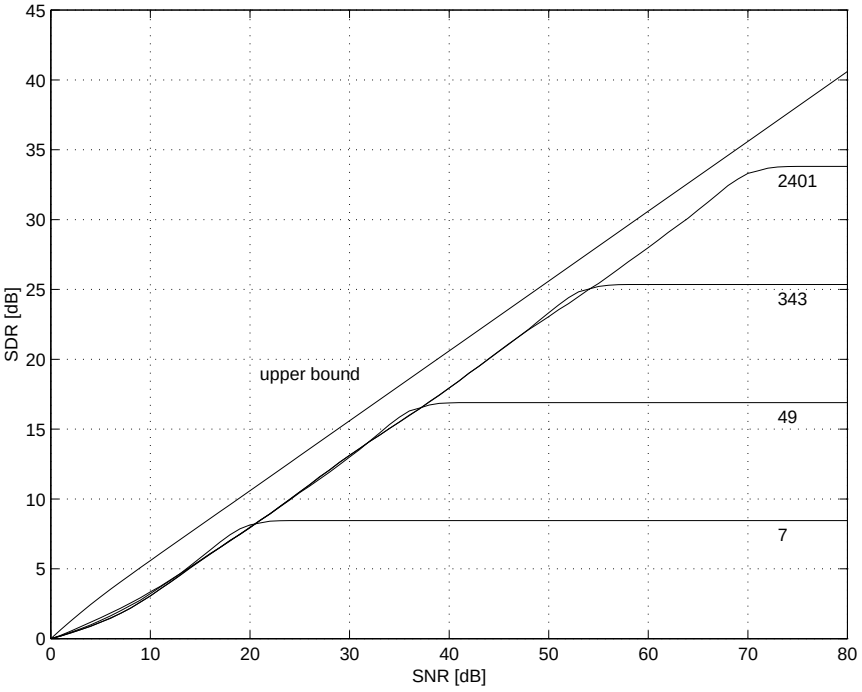


(b) MMSE decoder

Figure B-5: 3:1 bandwidth reduction using the three-dimensional Hilbert curve for uniform sources. A lattice vector quantizer is used for quantization (number of points of each case shown). Average SDRs are plotted as functions of the channel SNR.



(a) Inverse-mapping decoder



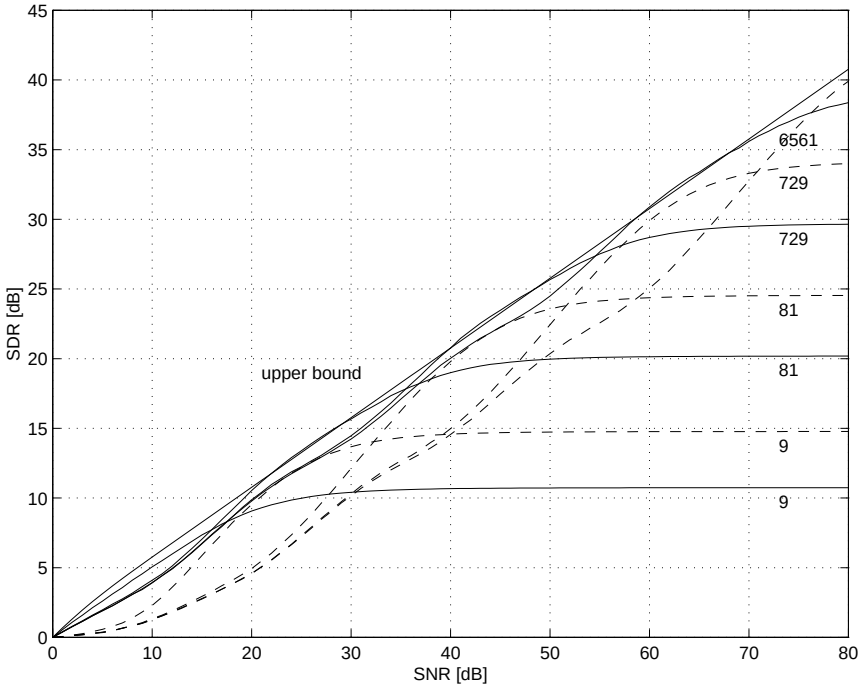
(b) MMSE decoder

Figure B-6: 2:1 bandwidth reduction using the hexagonal space-filling curve for uniform sources on $\mathcal{R}_V(A_2)$ A lattice vector quantizer is used for quantization (number of points of each case shown). SDRs are plotted as functions of the channel SNR.

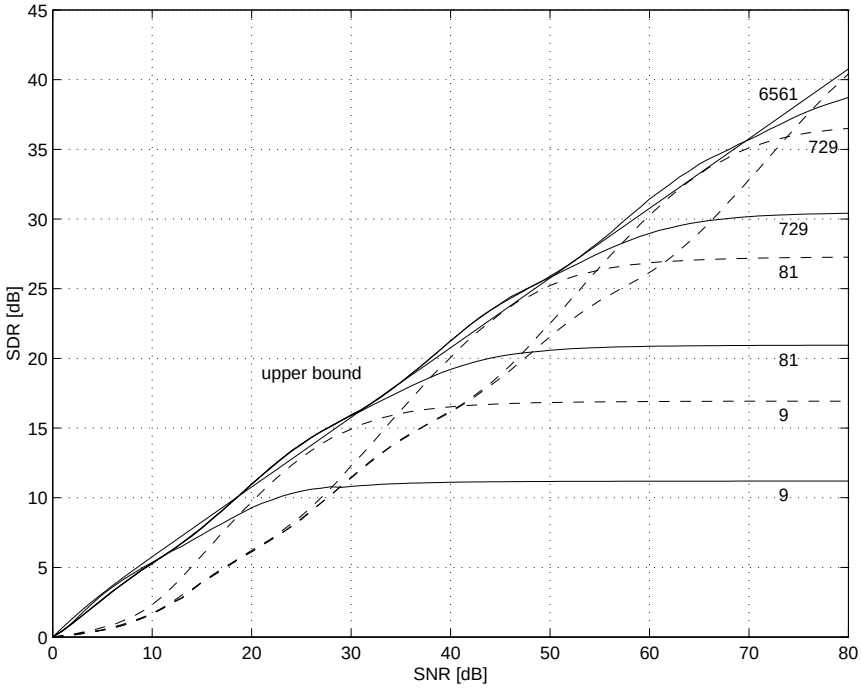
Appendix C

Simulation Results for Joint Coding Using Approximate Curves

In this appendix, we show Monte Carlo simulation results of 2:1 bandwidth reduction for uniform sources using several space-filling curves. Curves used are Peano curves in Figure 3-7 and Hilbert curves in Figure 3-8. Approximate curves for these curves are used for quantization. Two types of decoders, namely the inverse-mapping and MMSE decoders, are used for each case. Figures C-1–C-9 show these simulation results.

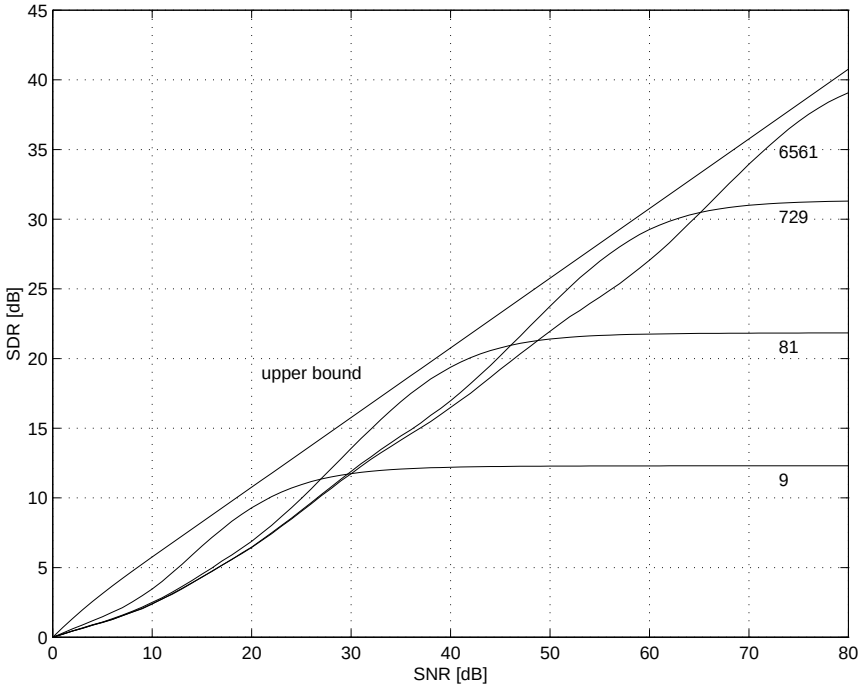


(a) Inverse-mapping decoder

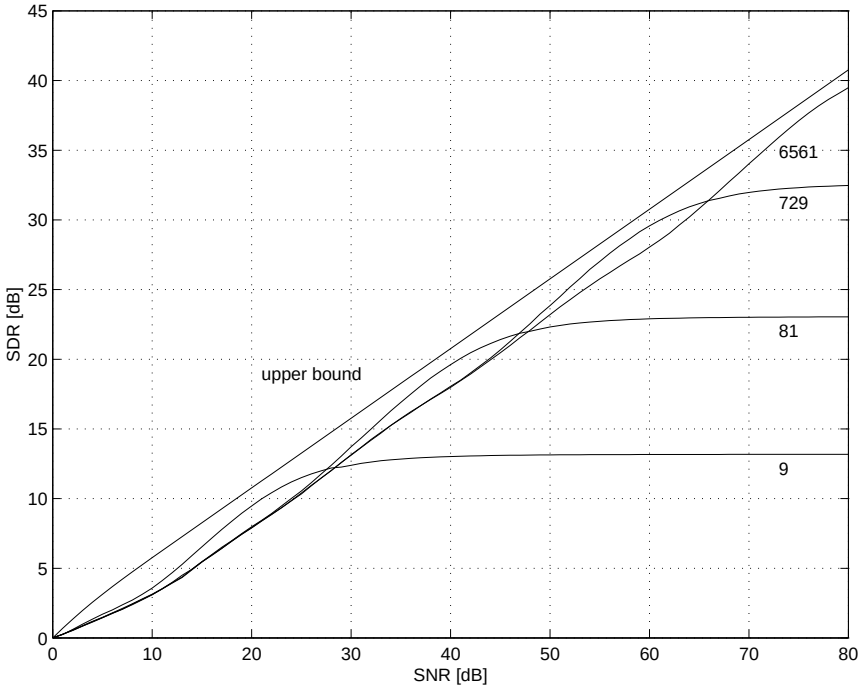


(b) MMSE decoder

Figure C-1: 2:1 bandwidth reduction using the Peano curve for uniform sources. Approximate curves are used for quantization (number of nodes of each curve shown). SDRs for W_1 (—) and W_2 (- -) are plotted as functions of the channel SNR.

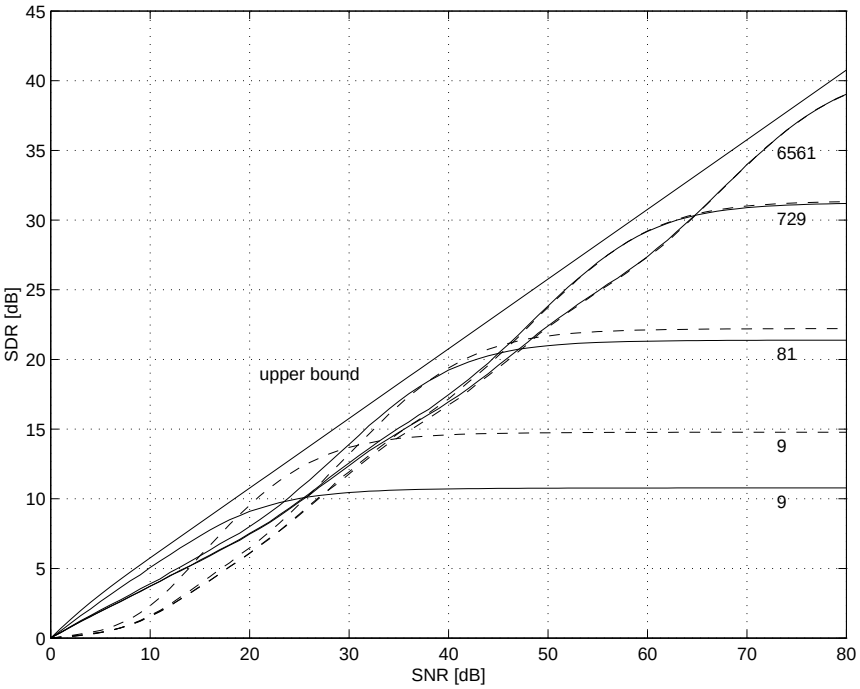


(a) Inverse-mapping decoder

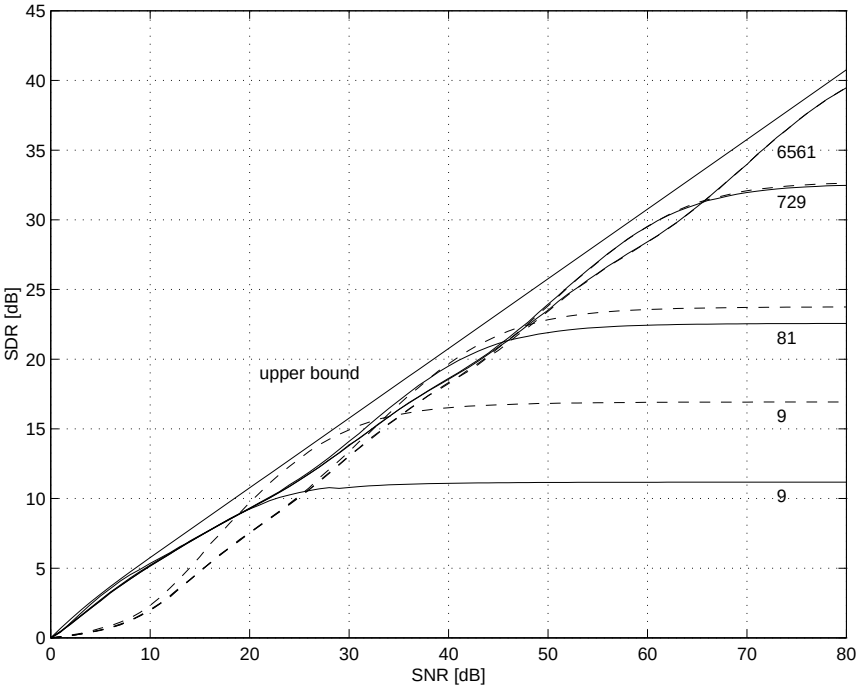


(b) MMSE decoder

Figure C-2: 2:1 bandwidth reduction using the Peano curve for uniform sources. Approximate curves are used for quantization (number of nodes of each curve shown). Average SDRs are plotted as functions of the channel SNR.

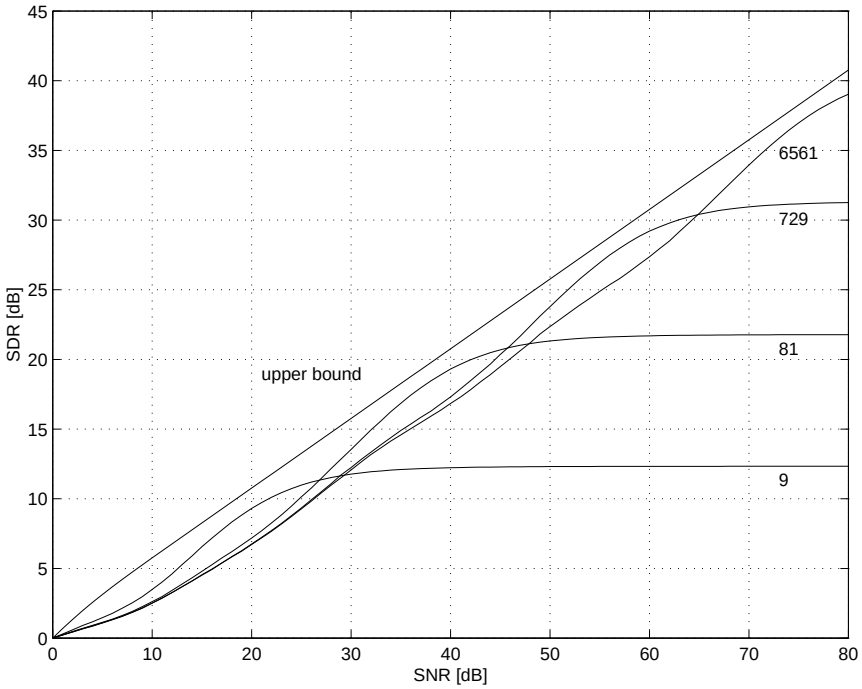


(a) Inverse-mapping decoder

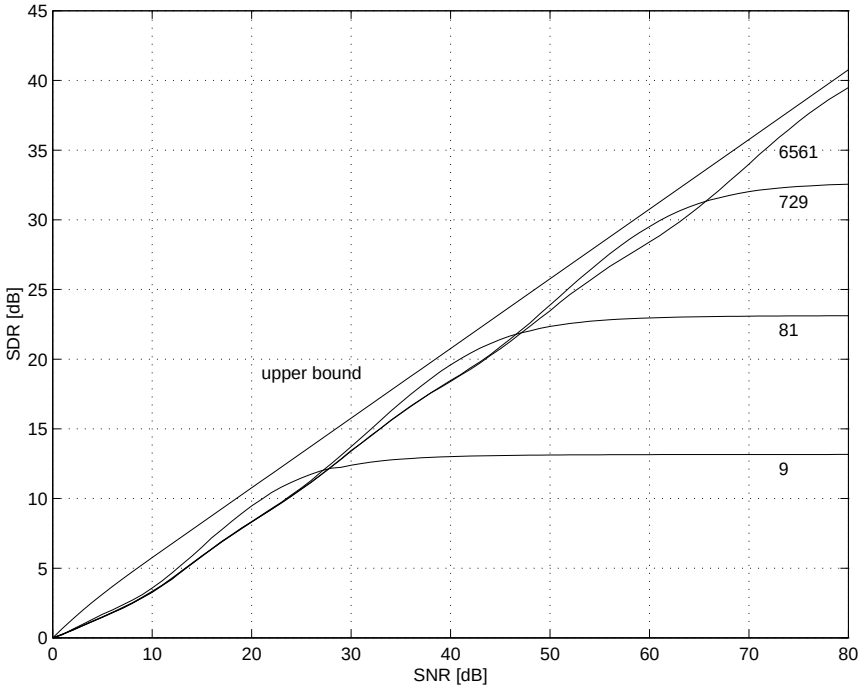


(b) MMSE decoder

Figure C-3: 2:1 bandwidth reduction using a Peano curve of the switch-back type in Figure 3-7 (b) for uniform sources. Approximate curves are used for quantization (number of nodes of each curve shown). SDRs for W_1 (—) and W_2 (- -) are plotted as functions of the channel SNR.

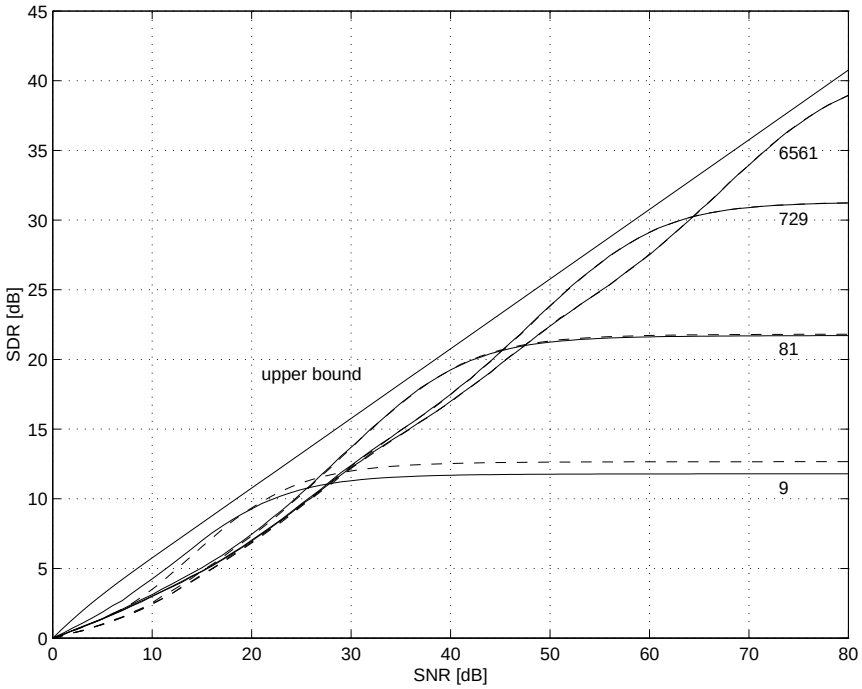


(a) Inverse-mapping decoder

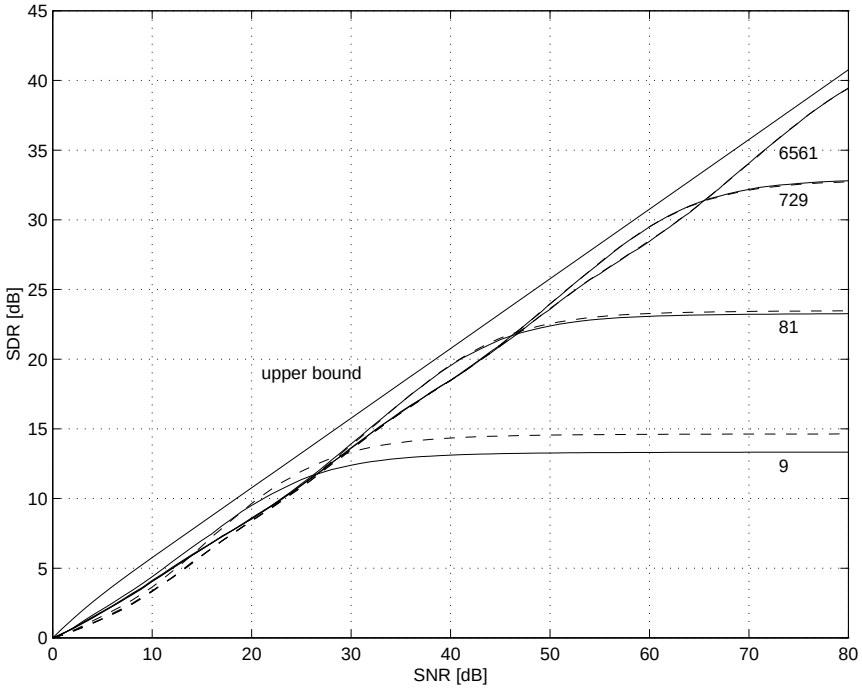


(b) MMSE decoder

Figure C-4: 2:1 bandwidth reduction using a Peano curve of the switch-back type in Figure 3-7 (b) for uniform sources. Approximate curves are used for quantization (number of nodes of each curve shown). Average SDRs are plotted as functions of the channel SNR.

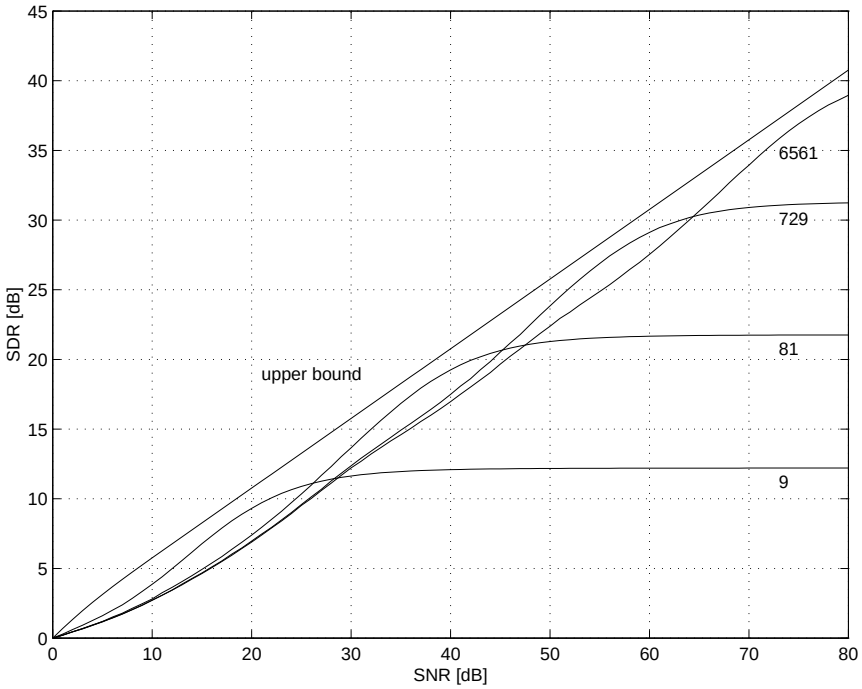


(a) Inverse-mapping decoder

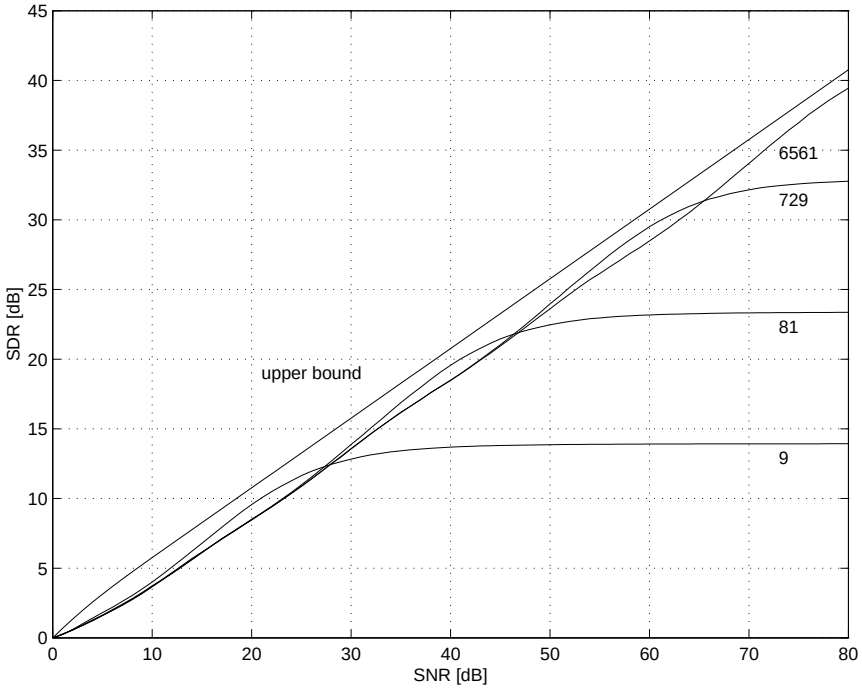


(b) MMSE decoder

Figure C-5: 2:1 bandwidth reduction using a Peano curve of the meander type in Figure 3-7 (c) for uniform sources. Approximate curves are used for quantization (number of nodes of each curve shown). SDRs for W_1 (—) and W_2 (- -) are plotted as functions of the channel SNR.

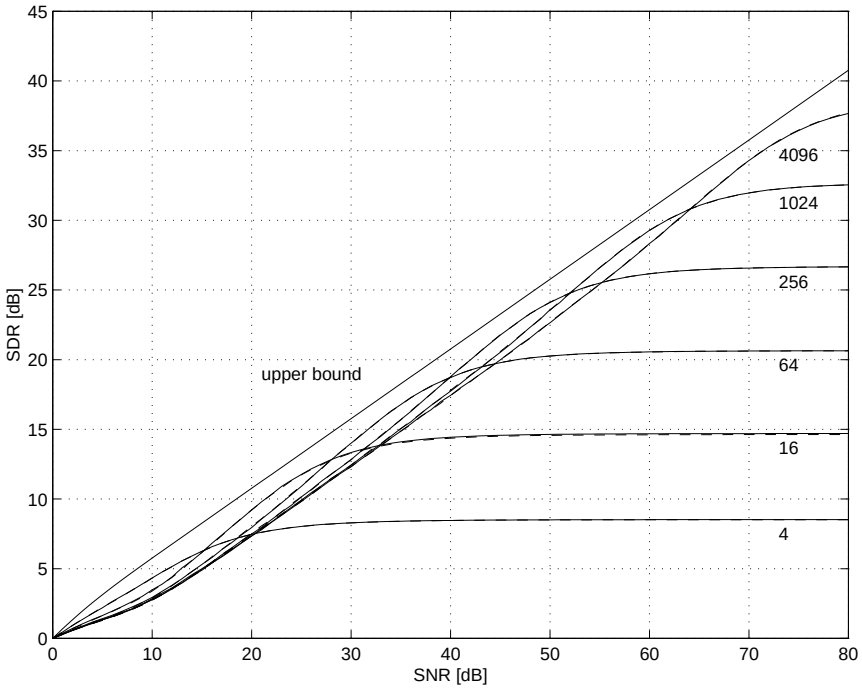


(a) Inverse-mapping decoder

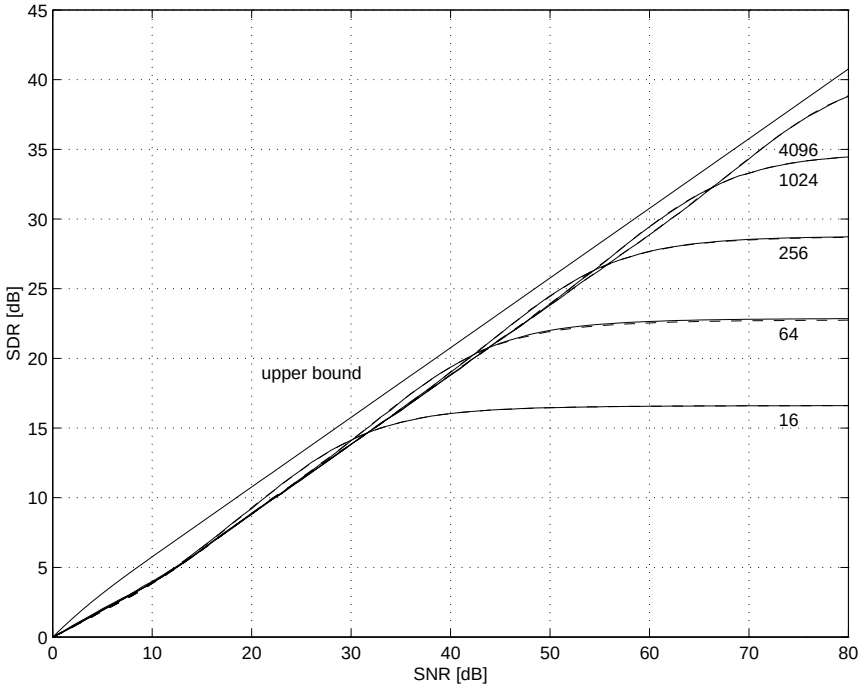


(b) MMSE decoder

Figure C-6: 2:1 bandwidth reduction using a Peano curve of the meander type in Figure 3-7 (c) for uniform sources. Approximate curves are used for quantization (number of nodes of each curve shown). Average SDRs are plotted as functions of the channel SNR.

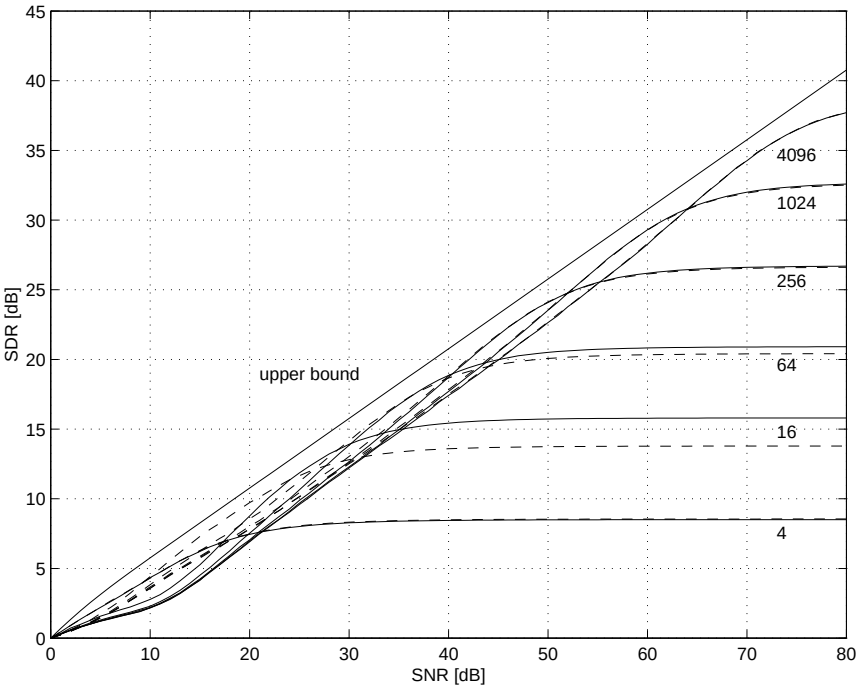


(a) Inverse-mapping decoder

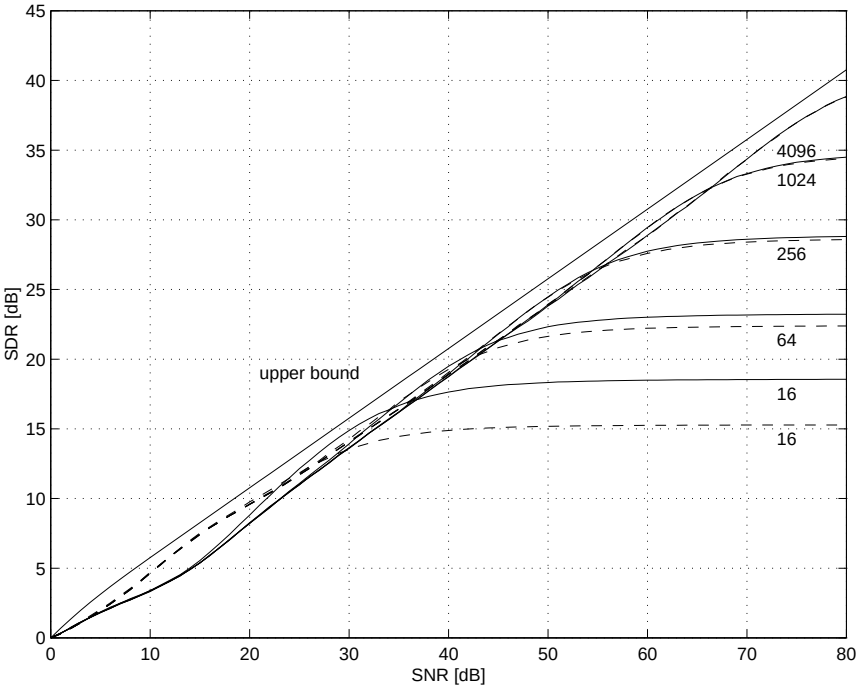


(b) MMSE decoder

Figure C-7: 2:1 bandwidth reduction using the Hilbert curve for uniform sources. Approximate curves are used for quantization (number of nodes of each curve shown). SDRs for W_1 (—) and W_2 (- -) are plotted as functions of the channel SNR.

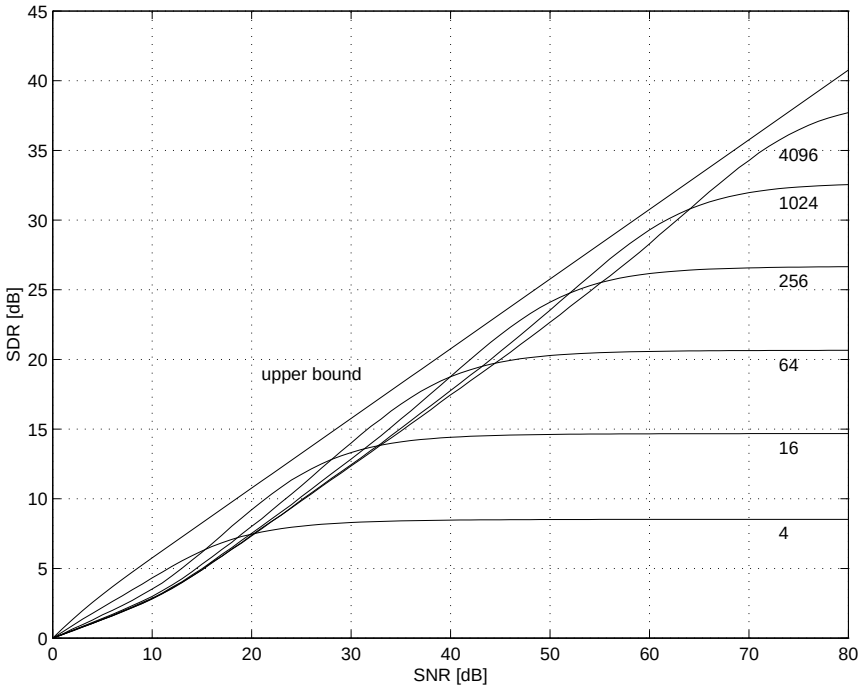


(a) Inverse-mapping decoder

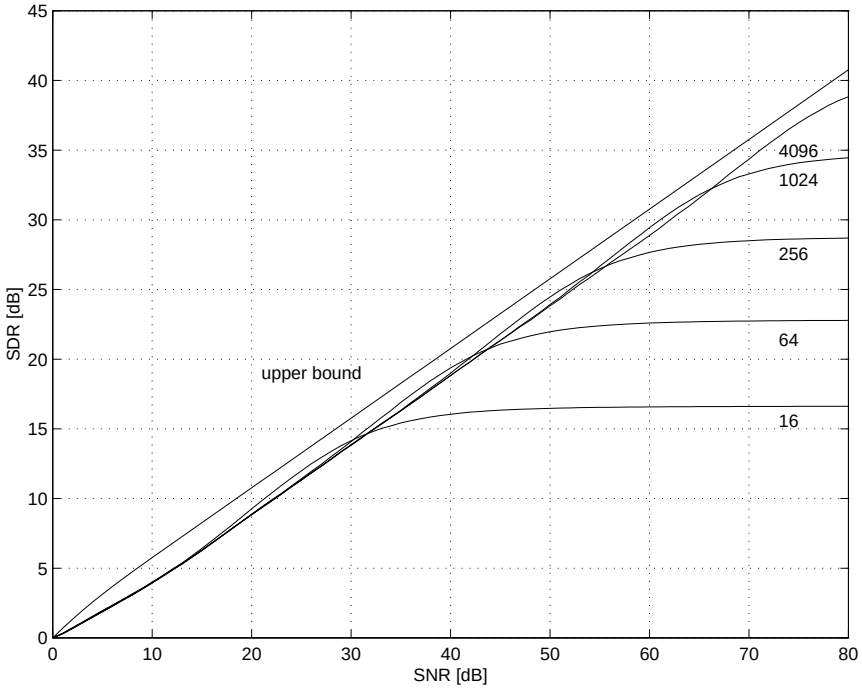


(b) MMSE decoder

Figure C-8: 2:1 bandwidth reduction using Moor's version of Hilbert's space-filling curve for uniform sources. Approximate curves are used for quantization (number of nodes of each curve shown). SDRs for W_1 (—) and W_2 (- -) are plotted as functions of the channel SNR.



(a) Inverse-mapping decoder



(b) MMSE decoder

Figure C-9: 2:1 bandwidth reduction using Moor's version of Hilbert's space-filling curve for uniform sources. Approximate curves are used for quantization (number of nodes of each curve shown). Average SDRs are plotted as functions of the channel SNR.

Bibliography

[1] S. M. Aji, G. B. Horn, and R. J. McEliece. On the convergence of iterative decoding on graphs with a single cycle. In *Proc. Int. Symp. Inform. Theory*, page 276, Cambridge, MA, August 1998.

[2] E. J. Anderson and P. Nash. *Linear Programming in Infinite-Dimensional Spaces: Theory and Applications*. Wiley, 1987.

[3] G. Battail, M. C. Decouvelaere, and P. Godlewski. Replication decoding. *IEEE Trans. Inform. Theory*, IT-25:332–345, May 1979.

[4] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo-codes. In *Proc. IEEE Int. Conf. Commun.*, pages 1064–1070, Geneva, Switzerland, May 1993.

[5] D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, Belmont, MA, 1997.

[6] T. Bially. Space-filling curves: Their generation and their application to bandwidth reduction. *IEEE Trans. Inform. Theory*, 15:658–664, November 1969.

[7] S.-Y. Chung and G. D. Forney, Jr. Normalized channels and approximation methods for low-density parity-check codes. *In preparation, 2000*.

[8] S.-Y. Chung, G. D. Forney, Jr., T. J. Richardson, and R. Urbanke. On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit. to appear in *IEEE Commun. Letters*.

[9] S.-Y. Chung, T. J. Richardson, and R. Urbanke. Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation. to appear in *IEEE Trans. Inform. Theory*.

[10] S.-Y. Chung and M. D. Trott. Joint source and channel coding using space-filling curves. *In preparation, 2000*.

[11] S.-Y. Chung and M. D. Trott. Joint source-channel coding using space-filling curves for bandwidth compression. In *Proc. Data Compression Conference*, page 536, Snowbird, Utah, March 1998.

[12] S.-Y. Chung, R. Urbanke, and T. J. Richardson. Gaussian approximation for sum-product decoding of low-density parity-check codes. In *Proc. Int. Symp. Inform. Theory*, page 318, Sorrento, Italy, June 2000.

[13] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, New York, 1991.

[14] I. Csiszár and J. Körner. *Information Theory: Coding Theorems for Discrete Memoryless Systems*. Academic Press, New York, 1981.

[15] P. Delsarte and P. Piret. Algebraic constructions of Shannon codes for regular channels. *IEEE Trans. Inform. Theory*, IT-28(4):593–599, 1982.

[16] H. El Gamal and A. R. Hammons, Jr. Analyzing the turbo decoder using the Gaussian approximation. In *Proc. Int. Symp. Inform. Theory*, page 319, Sorrento, Italy, June 2000.

[17] M. V. Eyuboglu and G. D. Forney, Jr. Trellis precoding: Combined coding, precoding and shaping for intersymbol interference channels. *IEEE Trans. Inform. Theory*, 38:301–314, March 1992.

[18] N. Favardin and V. A. Vaishampayan. Optimal quantizer design for noisy channels: An approach to combined source-channel coding. *IEEE Trans. Inform. Theory*, 33(6):827–838, November 1987.

[19] G. D. Forney, Jr. Codes on graphs: Normal realizations. to appear in *IEEE Trans. Inform. Theory*.

[20] G. D. Forney, Jr., F. Kschischang, and B. Marcus. Iterative decoding of tail-biting trellisses. In *Proc. Inform. Theory Workshop*, pages 11–12, San Diego, February 1998.

[21] G. D. Forney, Jr., M. D. Trott, and S.-Y. Chung. On the capacity of coset codes and multilevel coset codes. In *Proc. Int. Symp. Inform. Theory Appl.*, Victoria, B.C., Canada, September 1996.

[22] G. D. Forney, Jr., M. D. Trott, and S.-Y. Chung. Sphere-bound-achieving coset codes and multilevel coset codes. *IEEE Trans. Inform. Theory*, 46(3):820–850, May 2000.

[23] B. Frey. Turbo factor analysis. submitted to *Neural Computation*.

[24] A. Fuldseth. *Robust subband video compression for noisy channels with multilevel signaling*. PhD thesis, Dept. of Telecommunications, Norwegian University of Science and Technology, Trondheim, Norway, 1997.

[25] A. Fuldseth and T. A. Ramstad. Bandwidth compression for continuous-amplitude channels based on vector approximation to a continuous subset of the source signal space. In *Proc. ICASSP*, volume IV, pages 3093–3096, 1997.

[26] R. G. Gallager. Low-density parity-check codes. *IRE Trans. Inform. Theory*, IT-8:21–28, January 1962.

[27] R. G. Gallager. *Low-Density Parity-Check Codes*. MIT Press, Cambridge, MA, 1963.

[28] R. G. Gallager. *Information Theory and Reliable Communication*. Wiley, New York, 1968.

[29] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic, Boston, MA, 1992.

[30] T. J. Goblick, Jr. Theoretical limitation on the transmission of data from analog sources. *IEEE Trans. Inform. Theory*, 12:558–566, October 1965.

[31] C. Gotsman and M. Lindenbaum. Euclidean Voronoi labelling on the multidimensional grid. *Pattern Recognition Lett.*, 16:409–415, April 1995.

[32] C. Gotsman and M. Lindenbaum. On the metric properties of discrete space-filling curves. *IEEE Trans. Image Processing*, 5:794–797, May 1996.

[33] I. S. Gradshtein and I. M. Ryzhik. *Table of Integrals, Series, and Products*. Academic Press, New York, 1965.

[34] J. Hagenauer, E. Offer, and L. Papke. Iterative decoding of binary block and convolutional codes. *IEEE Trans. Inform. Theory*, 42:429–445, March 1996.

[35] C. R. Hartmann and L. D. Rudolph. An optimum symbol-by-symbol decoding rule for linear codes. *IEEE Trans. Inform. Theory*, IT-22:514–517, September 1976.

[36] D. Hilbert. Über die stetige abbildung einer linie auf ein flächenstück. *Math. Annln., 38*, pages 459–460, 1891.

[37] H. Imai and S. Hirakawa. A new multi-level coding method using error-correcting codes. *IEEE Trans. Inform. Theory*, 23(3):371–377, May 1977.

[38] F. Jelinek. *Probabilistic Information Theory*. McGraw-Hill, New York, 1968.

[39] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. submitted to *IEEE Trans. Inform. Theory*, 1998.

[40] C.-H. Lamarque and F. Robert. Image analysis using space-filling curves and 1D wavelet bases. *Pattern Recognition*, 29(8):1309–1322, 1996.

[41] K.-H. Lee. *Optimal linear coding for a multichannel system*. PhD thesis, Univ. of New Mexico, Albuquerque, NM, 1975.

[42] K.-H. Lee and D. P. Petersen. Optimal linear coding for vector channels. *IEEE Trans. Commun.*, COM-24(12):1283–1290, December 1976.

[43] A. Lempel and J. Ziv. Compression of two-dimensional data. *IEEE Trans. Inform. Theory*, 32(1):2–8, January 1986.

[44] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman. Analysis of low density codes and improved designs using irregular graphs. In *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing*, pages 249–258, 1998.

[45] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann. Practical loss-resilient codes. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing*, pages 150–159, 1997.

[46] D. J. C. MacKay and R. M. Neal. Near Shannon limit performance of low-density parity-check codes. *Electronics Letters*, 32:1645–1646, August 1996.

[47] B. B. Mandelbrot. *Fractals: Form, Chance, and Dimension.* Freeman, San Francisco, 1977.

[48] G. Mitchison and R. Durbin. Optimal numberings of an $N \times N$ array. *SIAM J. Alg. Disc. Meth.*, 7(4):571–582, October 1986.

[49] G. Peano. Sur une courbe qui remplit toute une aire plane. *Math. Annln., 36*, pages 157–160, 1890.

[50] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA, 1988.

[51] A. Pérez, S. Kamata, and E. Kawaguchi. Peano scanning of arbitrary size images. In *Proc. Int. Conf. Patt. Recogn.*, pages 565–568, 1992.

[52] R. J. Pilc. The transmission distortion of a source as a function of the encoding block length. *Bell Sys. Tech. J.*, 47(6):827–885, 1968.

[53] T. J. Richardson, A. Shokrollahi, and R. Urbanke. Design of capacity-approaching low-density parity-check codes. to appear in *IEEE Trans. Inform. Theory*.

[54] T. J. Richardson and R. Urbanke. The capacity of low-density parity-check codes under message-passing decoding. to appear in *IEEE Trans. Inform. Theory*.

[55] T. J. Richardson and R. Urbanke. Proof of the stability condition of density evolution. *in preparation*, 2000.

[56] T. J. Richardson and R. Urbanke. Thresholds for turbo codes. In *Proc. Int. Symp. Inform. Theory*, page 317, Sorrento, Italy, June 2000.

[57] P. Rusmevichientong and B. Van Roy. An analysis of turbo decoding with Gaussian priors. In *Advances in Neural Information Processing Systems*, 1999.

[58] H. Sagan. *Space-Filling Curves*. Springer-Verlag, New York, 1994.

[59] C. E. Shannon. A mathematical theory of communication. *Bell Sys. Tech. J.*, 27:379–423 and 623–656, July/Oct. 1948.

[60] C. E. Shannon. Communication in the presence of noise. *Proc. IRE*, pages 10–21, January 1949.

[61] A. Shokrollahi. New sequences of linear time erasure codes approaching the channel capacity. In *Proceedings of AAECC-13, Lecture Notes in Computer Science 1719*, pages 65–76, 1999.

[62] M. Sipser and D. A. Spielman. Expander codes. *IEEE Trans. Inform. Theory*, 42(6):1710–1722, November 1996.

[63] R. M. Tanner. A recursive approach to low complexity codes. *IEEE Trans. Inform. Theory*, 27:533–547, September 1981.

[64] S. ten Brink. Iterative decoding trajectories of parallel concatenated codes. In *3rd IEEE/ITG Conference on Source and Channel Coding*, Munich, Germany, January 2000.

[65] V. A. Vaishampayan. *Combined source-channel coding for bandlimited waveform channels*. PhD thesis, University of Maryland, 1989.

[66] A. J. Viterbi and J. K. Omura. *Principles of Digital Communication and Coding*. McGraw-Hill, 1979.

[67] D. Voorhies. Space-filling curves and a measure of coherence. In J. Arvo, editor, *Graphics Gems II*, pages 26–30. Academic, New York, 1991.

[68] U. Wachsmann and J. Huber. Power and bandwidth efficient digital communication using turbo codes in multilevel codes. *European Trans. Telecomm.*, 6(5), 1995.

[69] S. Wagon. *Mathematica in Action*. Freeman, 1991.

[70] Y. Weiss. Correctness of local probability propagation in graphical models with loops. to appear in *Neural Computation*, 2000.

[71] Y. Weiss and W. T. Freeman. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. In *Advances in Neural Information Processing Systems*, 1999.

[72] N. Wiberg. *Codes and decoding on general graphs*. PhD thesis, U. Linköping, Sweden, 1996.

[73] N. Wiberg, H.-A. Loeliger, and R. Kötter. Codes and iterative decoding on general graphs. *European Trans. Telecomm.*, 6:513–525, Sept./Oct. 1995.

[74] J. M. Wozencraft and I. M. Jacobs. *Principles of Communication Engineering*. Wiley, New York, 1965.

[75] A. D. Wyner and J. Ziv. On communication of analog data from a bounded source space. *Bell Sys. Tech. J.*, 48(10):3139–3172, December 1969.

[76] J. Ziv. The behavior of analog communication systems. *IEEE Trans. Inform. Theory*, 16(5):587–594, September 1970.