UNIVERSITY OF CALIFORNIA

Los Angeles

Constructions, applications, and implementations of low-density parity-check codes

A dissertation submitted in partial satisfaction of the

requirements for the degree Doctor of Philosophy

in Electrical Engineering

by

Christopher R. Jones

2003

The dissertation of Christopher R. Jones is approved.

Adnan Darwiche

Michael P. Fitz

Richard D. Wesel, Committee Co-Chair

John D. Villasenor, Committee Co-Chair

University of California, Los Angeles

2003

To friends and family

Contents

List of Figures

List of Tables

VITA

1995 B.S.E.E. University of California Los Angeles. Magna Cum Laude.

1996 M.S.E.E. University of California Los Angeles.

1997- 2002 VLSI System/ASIC Engineer, Broadcom Corporation

1995 - 1996, 2001 - 2003 Graduate Student Researcher, UCLA Electrical Engineering

2003 Ph.D. in Electrical Engineering, University of California Los Angeles.

PUBLICATIONS

B. Schoner, C. Jones, J. Villasenor, “Issues in wireless video coding using run-time-reconfigurable FPGAs,” Proceedings IEEE Symposium on FPGAs for Custom Computing Machines, Napa Valley, CA, USA, 19-21 April 1995.

J. Villasenor, R. Jain, B. Belzer, W. Boring, C. Chien, C. Jones, J. Liao, S. Molloy, S. Nazareth, B. Schoner, J. Short, “Wireless video coding system demonstration,” Proceedings. DCC ’95 Data Compression Conference, Snowbird, UT, USA.

J. Villasenor, C. Jones, B. Schoner, “Video Communications Using Rapidly Reconfigurable Hardware,” *IEEE Transactions on Circuits and Systems for Video Technology*, Dec. 1995. p.565-7.

J. Villasenor, B. Schoner, K.N.Chia, C. Zapata, H.J. Kim, C. Jones, S. Lansing, B. Mangione-Smith, “Configurable computing solutions for Automatic Target Recognition,” Proceedings IEEE Symposium on FPGAs for Custom Computing Machines, Napa Valley CA, USA, April 1996.

F. Lu, J. Min, S. Liu, K. Cameron, C. Jones, O. Lee, J. Li, A. Buchwald, S. Jantzi, C. Ward, K. Choi, J. Searle, H. Samueli, “A single-chip universal burst receiver for cable modem/digital cable-TV applications,” Custom Integrated Circuits Conference, 2000. CICC. Proceedings of the IEEE 2000 , 21-24 May 2000 Page(s): 311 -314

K. Lakovic, C. Jones, J. Villasenor, “Investigating Quasi Error Free (QEF) Operation with Turbo Codes,” Proceedings IEEE International Symposium on Turbo Codes, Brest, France, Sept 2000.

C. Jones, T. Tian, J. Villasenor, R. Wesel, “Robustness of LDPC Codes on Periodic Fading Channels,” Proceedings GlobeCom, Taipei, Taiwan, Nov 2002.

T. Tian, C. Jones, R. Wesel, J. Villasenor, “Construction or irregular LDPC codes with low error floors,” Proceedings ICC, Alaska, USA, May 2003.

C. Jones, A. Matache, T. Tian, J. Villasenor, R. Wesel, “Approximate-Min* Constraint Node Updating for LDPC Code Decoding,” Proceedings MILCOM 2003, Boston, MA.

C. Jones, E. Vallés, M. Smith, J. Villasenor, “The Universality of LDPC Codes on

Wireless Channels,” Proceedings MILCOM 2003, Boston, MA.

ABSTRACT OF THE DISSERTATION

Constructions, applications, and implementations of low-density parity-check codes

by

Christopher R. Jones

Doctor of Philosophy in Electrical Engineering

University of California, Los Angeles, 2003

Professor Richard D. Wesel, Co-Chair

Professor John D. Villasenor, Co-Chair

The work described in this thesis is related to the application and design of Low-Density Parity-Check (LDPC) Codes for wireless channels. Advances in code analysis and dramatic reductions in transistor sizing have promoted LDPC codes to the forefront of applicable forward error correction technologies. The problem of code construction has been addressed in this work and we have produced a rate-flexible reduced error floor LDPC matrix design methodology. En route to the proposal of a construction technique, the relationships between cycles, stopping sets, and codewords are described. A discussion of how these structures limit LDPC code performance under message passing decoding follows. A new metric called extrinsic message degree (EMD) measures cycle connectivity in bipartite graphs. Using an easily computed estimate of EMD, we propose a Viterbi-like algorithm that selectively avoids cycles and increases minimum stopping set size, which is closely related to minimum distance. This algorithm yields codes with error floors that are orders of magnitude below those of girth-conditioned codes. The resulting codes have good waterfall-region and error-floor performance over a wide range of code rates and block sizes.

Another main contribution of the thesis stems from analytic and simulation based results for LDPC codes on frequency selective channels under othogonal frequency divsion modulation and generalized Gaussian channels. A particular emphasis on the robustness of the codes in fading environments is made. A-posterior probability and minimum-mean-square-error successive-interference-cancellation detection techniques, with several variants of the latter, have been considered. Analysis and simulation of code performance in parameterized period-2 scalar fading channels, random period-p fading channels, partial band jamming channels, parameterized 2×2 quasi-static multi-input multi-output channels, and N$\times$N Rayleigh fast fading MIMO channels are reported.

Of course, deployment of LDPC coding techniques requires that attention be paid to decoder complexity and large scale integration design issues. In steps toward this end, a high-throughput digital decoder architecture and implementation has been produced. The implementation includes an algorithmic modification of constraint node update processing, as well as message passing data path considerations, and has been tested in a lab prototype using a field programmable gate array device.

Chapter 1

Introduction to Low-Density Parity-Check Coding

Iterative techniques that permit codes with very long block lengths to be decoded with a complexity that is nearly linear in the length of the code will enable next generation communication links to be ‘optimal’ in terms of the throughput they will support for a given received signal-to-noise ration.

Low-density parity-check (LDPC) codes were proposed by Gallager in the early 1960s [18]. The structure of Gallager’s codes (uniform column and row weight) led them to be called regular LDPC codes. Gallager provided simulation results for codes with block lengths on the order of hundreds of bits. However, these codes were too short for the sphere packing bound to approach Shannon capacity, and the computational resources for longer random codes were decades away from being broadly accessible.

Following the groundbreaking demonstration by Berrou *et al.* [4] of the impressive capacity-approaching capability of long random linear (turbo) codes, MacKay [29] re-established interest in LDPC codes during the mid to late 1990s. Luby *et al.* [28] formally showed that properly constructed irregular LDPC codes can approach capacity more closely than regular codes. Richardson, Shokrollahi and Urbanke [36] created a systematic method called density evolution to analyze and synthesize the *degree distribution* in asymptotically large random bipartite graphs under a wide range of channel realizations.

LDPC codewords are generated through the random linear superposition of basis vectors that define the code. If the binary Hamming distance between all combinations of codewords (the distance spectrum) is known, then analytic techniques for describing the performance of the codes in the presence of noise are available. However, in the case of LDPC codes (which are random linear codes), the problem of finding the distance spectrum of the code is intractable. Researchers instead resort to the use of Monte Carlo simulation in order to characterize the performance of various code constructions. Of particular interest is the performance of these codes at high signal to noise ratios (SNRs) where errors occur very rarely. Thorough characterization of a code in this region may require simulation of $10^{10} - 10^{12}$ code symbols. Therefore, throughputs on the order of 10^6 code symbols per second are required if the high SNR performance of a given code is to be resolved within a reasonable amount of time. In chapter 2 of this dissertation, an implementation that allows high speed Monte Carlo simulation of Low-Density Parity-Check (LDPC) codes is introduced. In particular, the belief propagation (BP) algorithm is carefully deconstructed to obtain a form that is compatible with integer implementation. Additionally, a parallel application of the integer BP algorithm constitutes the high throughput architecture that we present as a final result.

After the discussion of complexity reduced decoding techniques, chapter 3 turns to the problem of code (parity matrix) construction. The code construction portion of this work was performed jointly with Tao Tian, a fellow graduate student in UCLA Electical Engineering. In this work, the relationships between cycles, stopping sets, and codewords of the code parity check matrix are described. A discussion of how these structures limit LDPC code performance under message passing decoding follows. A new metric called extrinsic message degree (EMD) measures cycle connectivity in bipartite graphs. Using an easily computed estimate of EMD, we propose a Viterbi-like algorithm that selectively avoids cycles and increases the minimum stopping set size, which is closely related to minimum distance. This algorithm yields codes with error floors that are orders of magnitude below those of girth-conditioned codes. The resulting codes have good waterfall-region and error-floor performance over a wide range of code rates and block sizes.

With well constructed codes and high speed simulation methods at hand, we turn our attention to applications of LDPC codes and describe their performance on two distinct classes of wireless channels. We endeavor throughout chapters 4 and 5 to evidence the ‘Universality’ or ‘Robustness’ of LDPC codes under widely differing channelization scenarios. We confidently enter this limitless sea of channels due to a result from Root and Varaiya who proved the existence of codes that can communicate reliably over any member of a set of linear Gaussian channels where the mutual information level of each member exceeds a given threshold. In these chapters we show that Low-Density

Parity-Check (LDPC) codes are such codes and that their performance lies in close proximity to the Root and Varaiya capacity for a large family of scalar-input fading channels (chapter 4) and vector-input fading channels (chapter 5). Specifically, the robustness of LDPC codes in scalar fading channel is demonstrated in chapter 4 through the consistency of their mutual information performance across periodic fading profiles. To aid in the analysis of LDPC codes on these channels, density evolution has been adapted to the periodic fading case. This tool will be used both to design codes matched to specific channels and to determine the threshold of existing codes across parameterizations of periodic fading channels.

In chapter 5, analogous robustness properties are demonstrated on vector-input (matrix/MIMO) linear channels. As a special case, the 2x2 channel is investigated in detail. It is possible to characterize the mutual information level of any of the 2x2 channels via a single parameter (the eigenskew of the channel). Eigenskew in conjunction with a sampling of unitary 2x2 transforms allows us to examine the performance of an LDPC code across essentially all 2x2 channels. The more general NxN case is examined for fast Rayleigh fading and code performance is measured against ergodic Rayleigh capacities in this case. Chapter 5 also presents work, performed jointly with Adina Matache, that examines the problem of vector detection at a receiver. While the A-Posterior Probability (APP) detector (also sometimes refered to as the MAP detector) is known to be optimal from a BER point of view, it's complexity scales exponentially with the number of transmit antennas and the spectral efficiency of the chosen modulation. Other detection mechanisms can achieve performance to APP/MAP with less complexity and several such alternatives are presented.

1.1 A Brief Introduction to Low-Density Parity-Check Codes

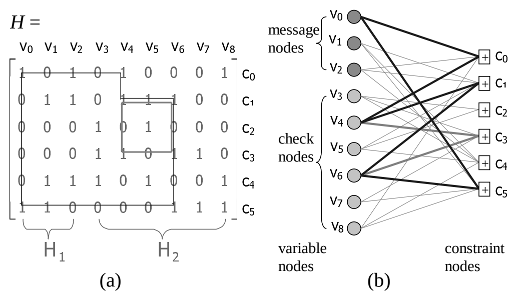


Figure 1.1. Matrix and graph descriptions of a (9, 3) code. A length 4 and a length 6 cycle are circumscribed with bold lines.

Like turbo codes, LDPC codes belong to the class of codes that are decodable primarily via *iterative* techniques. The demonstration of capacity approaching performance in turbo codes stimulated interest in the improvement of Gallager's original LDPC codes to the extent that the performance of these two code types is now comparable in AWGN. The highly robust performance of LDPC codes in other types of channels such as partial-band jamming, quasi-static multi-input multi-ouput (MIMO) Rayleigh fading, fast MIMO Rayleigh fading, and periodic fading is evidenced in [5] and [6].

LDPC codes are commonly represented as a bipartite graph (see Fig. 1.1b). In the graph, one set of nodes, the *variable* nodes, correspond to the codeword symbols and another set, the *constraint* nodes, represent the constraints that the code places on the variable nodes in order for them to form a valid codeword. *Regular* LDPC codes have bipartite graphs in which all nodes of the same type are of the same degree. A common example is the (3,6) regular LDPC code where all variable nodes have degree 3, and all constraint nodes have degree 6. The regularity of this code implies that the number of constraint nodes (which is the same as the number of parity check bits) equals exactly half the number of variable nodes such that the overall code is rate 1/2.

Mackay [29][30] has provided a diverse set of constructions for regular codes. Gallager [18] first showed that any particular random draw of a code from the (3,6) regular *ensemble* will result in a code whose error correcting performance lies asymptotically (in block length) close to the average performance of the ensemble. However, in the case of a randomly realized graph and decoding via the belief propagation (BP) algorithm [34][25] this statement is too strong. The reason follows from the well known fact that the BP algorithm, which is applied iteratively, explicitly assumes that the graph underlying any particular node is a tree. It hence performs non-optimally when the tree has shared vertices as will always be the case in random bi-partite graphs of interest. Practical LDPC codes are realized from an *expurgated* ensemble where effort is made to avoid special loop topologies during the construction of the code.

A major breakthrough in *irregular* LDPC code (having non-uniform column and row weight) design came with the invention of *density evolution* by Richardson, Shokrollahi, and Urbanke [36]. The authors showed that it is possible to predict a noise threshold below which a code realized from a given ensemble can be expected to converge to zero errors with high probability. A code ensemble is most often described via a pair of polynomials,

$$\lambda(x) = \sum_{\mathrm{i=1}}^{\mathrm{d_l max}} \lambda_i x^{i-1}, \rho(x) = \sum_{\mathrm{i=1}}^{\mathrm{d_r max}} \rho_i x^{i-1}. \tag{1.1}$$

The coefficients of $\lambda(x)$,($\rho(x)$) represent the fraction of *edges* emanating from variables (constraints) of various degree (in the bi-partite graph describing the code) as indicated by the powers of the place holding variables x^{deg-1}. For instance, the (3,6) regular code has $(\lambda(x), \rho(x)) = (x^2, x^5)$. Furthermore, since $\lambda(x)$, $\rho(x)$ are cumulative distribution functions (CDFs) $\lambda(1) = \rho(1) = 1$ always holds. Conversion between edge and node perspective is useful for defining the rate of the code in terms of a particular $(\lambda(x), \rho(x))$. Let E be the total number of edges in the graph, then $\frac{E\lambda_i}{i}$ equals the number of variable nodes with degree i. The rate of the code follows from the well known definition,

$$R = 1 - \frac{\sum \frac{\rho_i}{i}}{\sum \frac{\lambda_i}{i}} = 1 - \frac{\int \rho(x)}{\int \lambda(x)}. \tag{1.2}$$

The parity check matrix H of a linear binary (n, k) systematic code has dimension $(n - k) \times n$. The rows of H comprise the null space of the rows of the code's $k \times n$ generator matrix G. H can be written as,

$$H = \left[\begin{array}{cc} H_1 & H_2 \end{array} \right], \tag{1.3}$$

where H_1 is an $(n - k) \times k$ matrix and H_2 is an $(n - k) \times (n - k)$ matrix. H_2 is constructed to be invertible, so by row transformation through left multiplication with H_2^{-1}, we obtain a systematic parity check matrix H_{sys} that is range equivalent to H,

$$H_{sys} = H_2^{-1} H = \left[\begin{array}{cc} H_2^{-1} H_1 & I_{n-k} \end{array} \right]. \tag{1.4}$$

The left-hand portion of which can be used to define a null basis for the rows of H. Augmentation of the left-hand portion of the systematic parity check matrix H_{sys} with I_k yields the systematic generator matrix,

$$G_{sys} = \left[\begin{array}{cc} I_k & \left(H_2^{-1} H_1\right)^T \end{array} \right]. \tag{1.5}$$

The rows of G_{sys} span the codeword space such that $G_{sys} H^T = G_{sys} H_{sys}^T = 0$. It should be noted that although the original H matrix is sparse, neither H_{sys} nor G_{sys} is sparse in general. G_{sys} is used for encoding and the sparse parity matrix H is used for iterative decoding. A technique that manipulates H to obtain a nearly lower triangular form and allows essentially linear time (as opposed to the quadratic time due to a dense G matrix) encoding is available and was proposed by [37].

The matrix and graph descriptions of an $(n = 9, k = 3)$ code are shown in Fig. 1.1. Structures known as cycles, that affect decoding performance, are shown by (bold) solid lines in the figure. Although the relationship of graph topology to code performance in the case of a specific code is not fully understood, work exists [45] that investigates the effects of graph structures such as cycles, stopping sets, linear dependencies, and expanders.

Chapter 2

Belief Propogation Decoding via Reduced Complexity Techniques

In his original work, Gallager introduced several decoding algorithms. One of these algorithms has since been identified for general use in factor graphs and Bayesian networks [34] and is often generically described as Belief Propagation (BP). In the context of LDPC decoding, messages handled by a belief propagation decoder represent probabilities that a given symbol in a received codeword is either a one or a zero. These probabilities can be represented absolutely, or more compactly in terms of likelihood ratios or likelihood differences. The logarithmic operator can also be applied to either of these scenarios. Due to the complexity of the associated operator sets and wordlength requirements, the log-likelihood ratio form of the Sum-Product algorithm is the form that is best suited to VLSI implementation. However, this form still posses significant processing challenges as it employs a non-linear function that must be represented with a large dynamic range for optimal performance.

We note that even Full-BP algorithms suffer performance degradation as compared to the optimum ML decoder for a given code. This is due to the fact that bipartite graphs representing finite-length codes without singly connected nodes are inevitably non-tree-like. Cycles in bipartite graphs compromise the optimality of belief propagation decoders. The existence of cycles implies that the neighbors of a node are not in general conditionally independent (given the node), therefore graph separation does not hold and Pearl's polytree algorithm [34] (which is analogous to Full-BP decoding) inaccurately produces graph a-posteriori probabilities. Establishing the true ML performance of LDPC codes with length beyond a few hundred bits is generally viewed as an intractable problem. However, code conditioning techniques [45] (and chapter 3) can be used to mitigate the non-optimalities of iterative decoders and performance that approaches the Shannon capacity is achievable even with the presence of these decoding non-idealities. In what follows, we develop the Full-BP variable and constraint node update relations from initial principles. In succeeding sections, the constraint node update relations are modified to both reduce the quantity and the complexity of the required operations. We note that the proposed technique achieves both of these goals without incurring a measureable loss in performance.

2.1 Derivation of the Full BP Variable and Constraint Node Update Equations

At constraint node u_1 in Fig. 2.1 is,

$$v_A + v_B + v_C + v_E = 0 \tag{2.1}$$

The message we wish to compute is the one that is passed from u_1 back to v_A. This message can be computed as follows,

$$\begin{aligned} p_{A,u_1} =& P(v_A = 1, u_1|Y) \\ =& P(v_A = 1, v_A + v_B + v_C + v_E = 0|Y) \\ =& P(v_B + v_C + v_E = 1|Y) \\ =& p_B\left(1 - p_C\right)\left(1 - p_E\right) \\ & + p_C\left(1 - p_B\right)\left(1 - p_E\right) \\ & + p_E\left(1 - p_B\right)\left(1 - p_C\right) \\ & + p_B p_C p_E \\ =& \frac{1}{2} - \frac{1}{2}\prod_{i\in\{B,C,E\}}\left(1 - 2p_i\right) \end{aligned} \tag{2.2}$$

Messages that flow in the opposite direction adhere to a different update rule. Consider message $p_A^{(u_1)}$ is depicted in Fig. 2.2. This message can be computed in the following way,

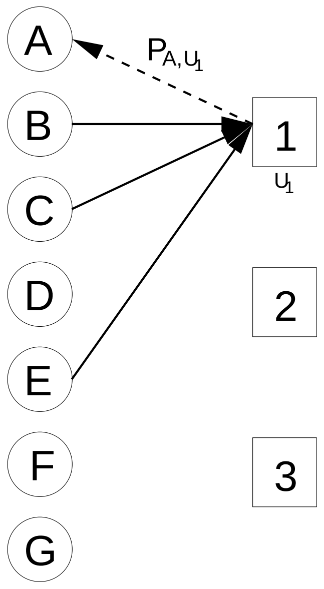


Figure 2.1. Partial bi-partite graph drawing that shows messages involved in the update of outgoing constraint message p_{A,u_1}.

$$\begin{aligned} p_A^{(u_1)} =& P\left(v_A = 1 | u_2, u_3, Y\right) \\ =& \frac{P\left(v_A = 1, u_2, u_3, Y\right)}{P\left(u_2, u_3, Y\right)} \\ =& \frac{P\left(v_A = 1, u_2, u_3, Y\right)}{P\left(v_A = 1, u_2, u_3, Y\right) + P\left(v_A = 0, u_2, u_3, Y\right)} \\ \approx& \frac{p_{A,u_2} p_{A,u_3}}{p_{A,u_2} p_{A,u_3} + \left(1 - p_{A,u_2}\right)\left(1 - p_{A,u_3}\right)} \end{aligned} \tag{2.3}$$

Where the quantity $P\left(v_A = 1, u_2, u_3 | Y\right)$ can be found as follows,

$$\begin{aligned} P\left(v_A = 1, u_2, u_3 | Y\right) =& P\left(\{v_A = 1, u_2 | Y\} \cap \{v_A = 1, u_3 | Y\}\right) \\ =& P\left(v_A = 1, u_2 | Y\right) P\left(v_A = 1, u_3 | Y\right) \\ =& p_{A,u_2} p_{A,u_3} \end{aligned} \tag{2.4}$$

However we note that the independence assumption made in the above relation does

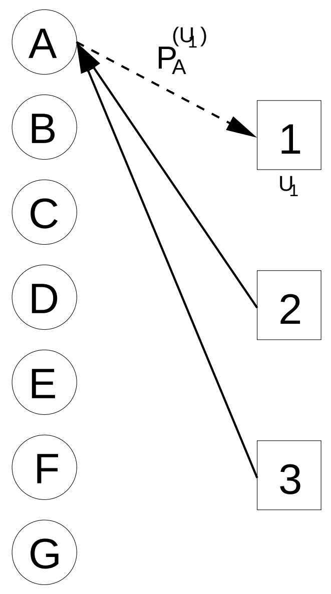


Figure 2.2. Partial bi-partite graph drawing that shows messages involved in the update of outgoing variable message $p_A^{(u_1)}$.

not hold in general. This is instead an approximate relationship due to cycle structures in the graph and is the reason for the approximation in the last line of (2.3).

Performing computations directly on probability measures is perfectly accurate, but not quite acceptable as both the variable and the constraint node operations require product operations that can be difficult to represent numerically. Instead, a transformation of the edge messages from the probability domain to the likelihood domain proceeds as follows. First consider the constraint update operations,

$$p_{A,u_1} = \frac{1}{2} - \frac{1}{2} \prod_{i \in \{B,C,E\}} (1 - 2p_i) \tag{2.5}$$

Which was defined previously. The likelihood ratio of this measure follows as,

$$\lambda_A = \frac{1-p_{A,u_1}}{p_{A,u_1}} = \frac{\frac{1}{2}+\frac{1}{2}\prod\limits_{i\in\{B,C,E\}}(1-2p_i)}{\frac{1}{2}-\frac{1}{2}\prod\limits_{i\in\{B,C,E\}}(1-2p_i)}$$
$$\overset{\text{AssumeDeg3}}{\Rightarrow} \frac{1-p_B-p_C-2p_Cp_B}{p_B+p_C-2p_Cp_B} = \frac{1+\left(\frac{1-p_C}{p_C}\right)\left(\frac{1-p_B}{p_B}\right)}{\left(\frac{1-p_C}{p_C}\right)+\left(\frac{1-p_B}{p_B}\right)} = \frac{1+\lambda_B\lambda_C}{\lambda_B+\lambda_C} \tag{2.6}$$

Then consider the following two definitions,

$$\ln\lambda_A = \ln\frac{1+\lambda_B\lambda_C}{\lambda_B+\lambda_C}\ \tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}} \tag{2.7}$$

Which when applied together produce the following relationship,

$$\tanh\left(\frac{1}{2}\ln\lambda_A\right) = \frac{1-e^{\frac{\lambda_B+\lambda_C}{1+\lambda_B\lambda_C}}}{1+e^{\frac{\lambda_B+\lambda_C}{1+\lambda_B\lambda_C}}} = \frac{(1-\lambda_B)(1-\lambda_C)}{(1+\lambda_B)(1+\lambda_C)} = \underbrace{\tanh\left(\frac{1}{2}\ln\lambda_B\right)\tanh\left(\frac{1}{2}\ln\lambda_C\right)}_{\text{Degree}-3\text{Case}} \tag{2.8}$$

for the degree-3 case and can be expressed as,

$$\tanh\left(\frac{1}{2}\ln\lambda_A\right) = \prod_{i=1}^{d_c-1}\tanh\left(\frac{1}{2}\ln\lambda_i\right) \tag{2.9}$$

In general. Going back to the variable node side of the graph, we restate the update relation,

$$p_A^{(u_1)} = \frac{p_{A,u_2}p_{A,u_3}}{p_{A,u_2}p_{A,u_3} + (1-p_{A,u_2})(1-p_{A,u_3})} \tag{2.10}$$

Likelyhood ratios applied here produce,

$$\lambda_A^{(u_1)} = \frac{1-p_A^{(u_1)}}{p_A^{(u_1)}} = \frac{(1-p_{A,u_2})(1-p_{A,u_3})}{p_{A,u_2}p_{A,u_3}} = \lambda_{A,u_2}\lambda_{A,u_3} \tag{2.11}$$

The logarithm of this form for the degree-3 case simply follows as,

$$\ln \lambda_A^{(u_1)} = \ln \lambda_{A,u_2} + \ln \lambda_{A,u_3} \tag{2.12}$$

and in general,

$$\ln \lambda_A^{(u_1)} = \sum_{i=1}^{dv-1} \ln \lambda_{A,i} \tag{2.13}$$

2.2 The Approximate-Min*-BP technique

This section is presented in several parts. In the first part, our modified version of Full-BP is introduced and its performance is contrasted to that of Full-BP. Next, we provide a discussion of the steps taken in the derivation of this technique. At the end of this section a complexity comparison between the proposed technique, Full-BP, and a previously proposed reduced complexity technique [8] is provided. Section 2.3 gives a discussion of finite precision issues and provides performance data for several quantization schemes. Section 2.4 describes the LDPC codec that has been developed at UCLA based on the constraint update technique that is presented in the chapter.

Before describing the technique, we introduce notation that will be used in the remainder of the chapter. On the variable node (left-hand) side of the bi-partite graph, u messages arrive and V messages depart. Both of these messages are log-likelihoods (e.g. $\ln \lambda$) as described in the previous section. At the constraint node (right-hand) side of the graph v messages arrive and U messages depart. All four message types are actually log-likelihood ratios (LLRs). For instance, a message v arriving at a constraint node is actually a shorthand representation for $v = ln(p(v = 0)/p(v = 1))$.

The constraint node *a-posteriori probability*, or U^{APP}, is defined as the constraint node message determined by the d_c variable messages that arrive at a constraint node of degree-d_c. The notation $U_j = U^{APP} \backslash v_j$ denotes the outgoing constraint message determined by all incoming edges with the exception of edge v_j. Message v_j represents *intrinsic* information that is left purposefully absent in the *extrinsic* message computation $U_j = U^{APP} \backslash v_j$. Our algorithm (called Approximate-Min*-BP, or A-Min*-BP) updates constraint messages as follows,

```
initialize $\left\{ v_{\min} = \min_{j=1..d_c} \left( |v_j| \right), \delta^0 = \infty \right\}$
for k =1 $\ldots d_c$
    if $(k \neq min)$
        $\delta^k = \left| \Lambda^{BP^*} \left( \delta^{k-1}, v_k \right) \right|$
    else:
        $\delta^k = \delta^{k-1}$
end
$\delta^{APP \backslash v_{\min}} = \delta^{d_c}$
$\delta^{APP} = \left| \Lambda^{BP^*} \left( \delta^{APP \backslash v_{\min}}, v_{\min} \right) \right|$
$\sigma^{APP} = \prod_{j=1}^{d_c} \operatorname{sgn}(v_j)$
```

where δ^k is a storage variable, and Λ^{BP^*} will be defined shortly. Constraint message updates are found by applying the following additional operations on the above quantities.

$$\underset{j=\{1..d_c\}\backslash \min}{U_j} = \operatorname{sgn}(v_j)\operatorname{sgn}(\sigma^{APP})\delta^{APP}$$
$$U_{\min} = \operatorname{sgn}(v_{\min})\operatorname{sgn}(\sigma^{APP})\delta^{APP\backslash v_{\min}}$$

The above constraint node update equations are novel and will be described in further detail. Variable node updating in our technique is the same as in the case of Full-BP. Extrinsic information is similarly described as before via $V_j = V^{APP}\backslash u_j$, however the processing required to achieve these quantities is much simpler,

$$V^{APP} = \sum_{j=0}^{d_v} u_j \qquad \underset{j=1..d_v}{V_j} = V^{APP} - u_j, \tag{2.14}$$

where d_v is the variable node degree.

2.2.1 Derivation and complexity of the A-Min*-BP technique

Derivation of the Approximate-Min*-BP constraint node update begins with the so called 'Log-Hyperbolic-Tangent' definition of BP constraint updating. In the equation below, sign and magnitude are separable since the sign of LnTanh(x) is determined by the sign of x.

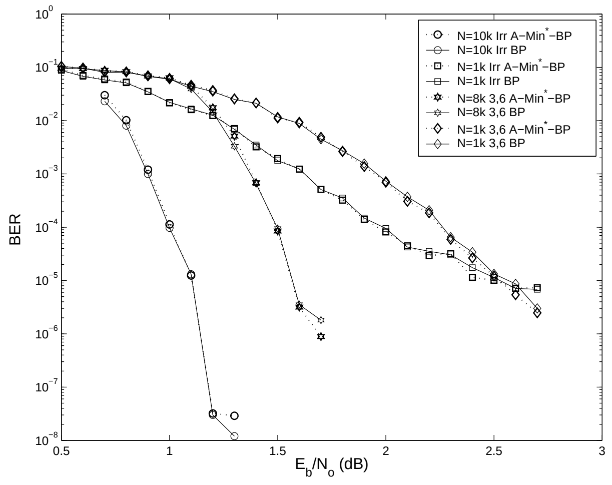


Figure 2.3. A-Min*-BP decoding compared to Full-BP decoding for four different rate 1/2 LDPC codes. Length 10k irregular max left degree 20, length 1k irregular max left degree 9, length 8k regular (3,6), length 1k regular (3,6). The proposed algorithm incurs negligible performance loss. Note that rate 1/2 BPSK constrained capacity is 0.18 dB (E_b/N_o). All simulations were performed with the same initial random seed.

$$U^{APP} = \left[\prod_{j=1}^{d_c} \operatorname{sgn}(v_j)\right] \ln \left(\frac{1 + e^{-\sum_{j=1}^{d_c} \ln\left(\frac{1+e^{-|v_j|}}{1-e^{-|v_j|}}\right)}}{1 - e^{-\sum_{j=1}^{d_c} \ln\left(\frac{1+e^{-|v_j|}}{1-e^{-|v_j|}}\right)}} \right). \qquad (2.15)$$

This equation is highly non-linear and warrants substantial simplification before mapping to hardware. To begin, the above computation can be performed by first considering the *inner* recursion in (2.15),

$$\sum_{j=1}^{d_c} \ln\left(\frac{1+e^{-|v_j|}}{1-e^{-|v_j|}}\right). \tag{2.16}$$

A total of d_c table look-ups to the function $\Lambda^{lnBP} = \ln\left(\frac{1+e^{-\left|v_j\right|}}{1-e^{-\left|v_j\right|}}\right)$ followed by $d_c - 1$ additions complete the computation in (2.16). Furthermore, the linearity of the inner recursion allows intrinsic variable values to be 'backed-out' of the total sum before d_c *outer* recursions are used to form the d_c extrinsic outputs. To summarize, computation of all d_c extrinsic values (in (2.15)) follow from d_c table look-ups, $d_c - 1$ additions, d_c subtractions, and a final d_c table look-ups. The cost of computing the extrinsic sign entails $d_c - 1$ exclusive-or operations to form the APP extrinsic sign, followed by d_c incremental exclusive-or operations to back-out the appropriate intrinsic sign to form each final extrinsic sign.

Variable node computation (2.14) is more straightforward. However, a possible alternative to (2.14) is given in [7] where it is noted that codes lacking low degree variable nodes experience little performance loss due to the replacement of V_j with V^{APP}. However, codes that maximize rate for a given noise variance in an AWGN channel generally have a large fraction of degree-2 and degree-3 variable nodes [36]. Low degree nodes are substantially influenced by any edge input and V^{APP} may differ significantly from corresponding properly computed extrinsic values. We have found experimentally that using V^{APP} alone to decode capacity approaching codes degrades performance by one dB of SNR or more.

We continue toward the definition of an alternative constraint update recursion by rearranging (2.15) for the $d_c = 2$ case,

$$\operatorname{sgn}(v_1)\operatorname{sgn}(v_2)\ln\left(\frac{1+e^{|v_1|+|v_2|}}{e^{|v_1|}+e^{|v_2|}}\right)=\ln\left(\frac{1+e^{v_1+v_2}}{e^{v_1}+e^{v_2}}\right). \tag{2.17}$$

Two applications of the Jacobian logarithmic identity ($\ln(e^a+e^b)=\max(a,b)+\ln\left(1+e^{-|a-b|}\right)$) [14] result in the Min* recursion that is discussed in the rest of the chapter,

$$\Lambda^{BP^*}(v_1,v_2)=\operatorname{sgn}(v_1)\operatorname{sgn}(v_2)\left(\begin{array}{l}\min\left(|v_1|,|v_2|\right)\\ +\ln\left(1+e^{-(|v_1|+|v_2|)}\right)\\ -\ln\left(1+e^{-||v_1|-|v_2||}\right)\end{array}\right). \tag{2.18}$$

Note that (2.18) is not an approximation. It is easy to show that d_c-1 recursions on Λ^{BP^*} yield exactly U^{APP} in equation (2.15). Furthermore, the function $\ln\left(1+e^{-|x|}\right)$ ranges over $(ln(2),0)$ which is substantially more manageable than the range of the function Λ^{LnBP}, $Range\left(\ln\left(\frac{1+e^{-|x|}}{1-e^{-|x|}}\right)\right)=(\infty,0)$ from a numerical representation point of view. However, the non-linearity of the recursion (2.18) implies that updating all extrinsic information at a constraint node requires $d_c\left(d_c-1\right)$ calls to Λ^{BP^*}. This rapidly becomes more complex than the $2d_c$ look-up operations (augmented with $2d_c-1$ additions) required to compute all extrinsic magnitudes based on the form in (2.15). Again, in this earlier case intrinsic values can be 'backed-out' of a single APP value to produce extrinsic values.

Instead of using the recursion in (2.18) to implement Full-BP we propose that this recursion be used to implement an approximate BP algorithm to be referred to as *Approximate-Min*-BP* (A-Min*-BP). The algorithm works by computing the proper extrinsic value for the minimum magnitude (least reliable) incoming constraint edge

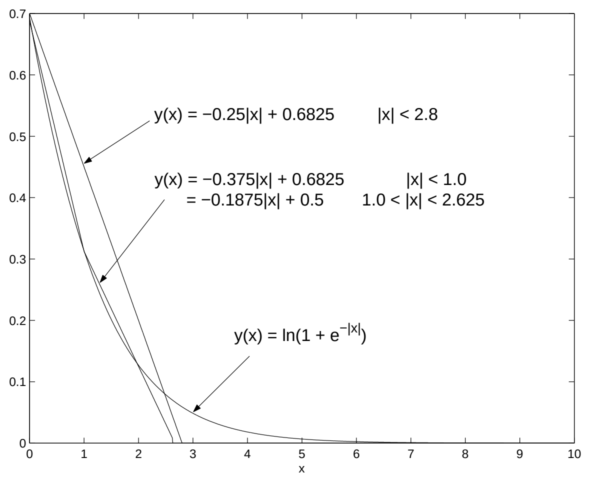


Figure 2.4. Non-linear function (Λ^{BP^*}) at the core of proposed algorithm and single / double line approximations to the function that can easily be implemented in combinatorial logic.

and assigning the U^{APP} magnitude in conjunction with the proper extrinsic sign to all other edges.

To provide intuition as to why this hybrid algorithm yields good performance, note first that a constraint node represents a single linear equation and has a known 'solution' if no more than one input variable is unknown. Consider the following two scenarios. First, if a constraint has more than one unreliable input, then all extrinsic outputs are unreliable. Second, if a constraint has exactly one unreliable input, then this unknown input can be solved for based on the extrinsic reliability provided by the 'known' variables. In this second case all other extrinsic updates are unreliable due to the contribution of the unreliable input. The approximation in the suggested algorithm assigns less accurate magnitudes to would-be unreliable extrinsics, but for the least reliable input preserves exactly the extrinsic estimate that would be produced by

Full-BP.

We next show that U^{APP} always underestimates extrinsics. Here the notation U_{mn} represents the extrinsic information that originates at constraint node m and excludes information from variable node n. Rearrangement of (2.15) (with standard intrinsic/extrinsic notation included [9]) yields the following,

$$\left|U^{APP}\right| = \ln \frac{1 + \prod_{n' \in N(m)} \frac{1-e^{-\left|v_{mn'}\right|}}{1+e^{-\left|v_{mn'}\right|}}}{1 - \prod_{n' \in N(m)} \frac{1-e^{-\left|v_{mn'}\right|}}{1+e^{-\left|v_{mn'}\right|}}} \tag{2.19}$$

$$\frac{1-e^{-\left|U^{APP}\right|}}{1+e^{-\left|U^{APP}\right|}} = \left(\prod_{n' \in N(m)\backslash n} \frac{1-e^{-\left|v_{mn'}\right|}}{1+e^{-\left|v_{mn'}\right|}}\right)\left(\frac{1-e^{-\left|v_{mn}\right|}}{1+e^{-\left|v_{mn}\right|}}\right). \tag{2.20}$$

Note first that the function $g(x) = \frac{1-e^{-|x|}}{1+e^{-|x|}}$ (a product of which comprises the RHS of (2.20)) ranges over $(0, 1)$ and is non-decreasing in the magnitude of x. The first (parenthesized) term on the right-hand side of (2.20) equals the extrinsic value U_{mn} under the operator $g(\cdot)$, i.e. $g(U_{mn})$. The second term scales this value by the intrinsic reliability $g(v_{mn})$. Hence, the monotonicity and range of $g(x)$ ensure that $\left|U^{APP}\right| < |U_{mn}|$. We provide the inverse function, $g^{-1}(x) = \ln \frac{1+x}{1-x}$, for reference.

Underestimation in A-Min*-BP is curtailed by the fact that the minimum reliability $g(v_{\min})$ dominates the overall product that forms U^{APP}. This term would have also been included in the outgoing extrinsic calculations used by Full-BP for all but the least reliable incoming edge. The outgoing reliability of the minimum incoming edge incurs no degradation due to underestimation since the proper extrinsic value is explicitly calculated. Outgoing messages to highly reliable incoming edges suffer little from underestimation since their corresponding intrinsic $g(v_{mn})$ values are close to

	Full-BP	A-Min*-BP	Offset-Min-BP
Table LookUps	$2d_c$	$d_c - 1$	0
Comparisons	0	$d_c - 1$	$2d_c - 3$
Additions	$2d_c - 1$	0	d_c
XORs	$2d_c - 1$	$2d_c - 1$	$2d_c - 1$
Tot Table Lookups	80000	35000	0
Tot Comparisons	0	35000	65000
Tot Additions	75000	0	40000
Tot XORs	75000	75000	75000
Tot Ops	230,000	145,000	180,000
Performance	Reference	No Loss	0.1dB Loss

Table 2.1. Complexity comparison for three constraint update techniques. Full-BP and A-Min*-BP have essentially the same performance. The simplest technique, Offset-Min-BP, experiences about a 0.1dB loss [7]. Numerical values are shown for a rate 1/2 code with: $n - k = 5000$**, and average right degree** d_c**=8.**

one. The worst case underestimation occurs when two edges ‘tie’ for the lowest level of reliability. In this instance the dominant term in (2.20) is squared. An improved version of A-Min*-BP would calculate exact extrinsics for the two smallest incoming reliabilities. However, the results in Fig. 2.3, where the algorithm (using floating point precision) is compared against Full-BP (using floating point precision) for short and medium length regular and irregular codes, indicate that explicit extrinsic calculation for only the minimum incoming edge is sufficient to yield performance that is essentially indistinguishable from that of Full-BP.

The proposed algorithm is similar to the Offset-Min-BP algorithm of [8] where the authors introduce a scaling factor to reduce the magnitude of extrinsic estimates produced by Min-BP. The Min-BP algorithm finds the magnitude of the two least reliable edges arriving at a given constraint node (which requires $d_c - 1$ comparisons followed by an additional $d_c - 2$ comparisons). The magnitude of the least reliable edge is assigned to all edges except the edge from which the least reliable magnitude came (which is assigned the second least reliable magnitude). For all outgoing edges, the proper extrinsic sign is calculated. As explained in [9] these outgoing magnitudes overestimate the proper extrinsic magnitudes because the constraint node update equation follows a product rule (2.20) where each term lies in the range $(0, 1)$. The Min-BP approximation omits all but one term in this product. To reduce the overestimation, an offset (or scaling factor) is introduced to decrease the magnitude of outgoing reliabilities. The authors in [8] use density evolution to optimize the offset for a given degree distribution and SNR. The optimization is sensitive to degree sequence selection and also exhibits SNR sensitivity to a lesser extent. Nevertheless, using optimized parameters, performance within 0.1 dB of Full-BP performance is possible.

By way of comparison, A-Min*-BP improves performance over Min-BP because the amount by which U^{APP} underestimates a given extrinsic is less than the amount by which Min-BP overestimates the same extrinsic. Specifically, the former underestimates due to the inclusion of one extra term in the constraint node product while the latter overestimates due to the exclusion of all but one term in the product. A direct comparison to Offset-Min-BP is more difficult. However, a simple observation is that in comparison to Offset-Min-BP, A-Min*-BP is essentially ‘self-tuning’.

The range and shape of the non-linear portion (Λ^{BP^*}) of the A-Min*-BP computation are well approximated using a single, or at most a 2-line, piecewise linear fit, as shown in Fig. 2.4. All of the fixed precision numerical results to be presented in section 2.3 use the 2-line approximation (as do the floating point results in Fig. 2.3). Hence, the entire constraint node update is implemented using only shift and add computations, no look-ups to tables of non-linear function values are actually required.

The cost of constraint node updating for Full-BP (implemented using (2.15)), A-Min*-BP, and Offset-Min-BP are given in Table 2.1. The latter two algorithms have similar cost with the exception that $d_c - 1$ table look-up operations in A-Min*-BP are replaced with d_c additions in Offset-Min-BP (for offset adjustment). Note that use of a table is assumed for the representation of Λ^{LnBP}. While Λ^{BP^*} is well approximated using a two line piecewise fit employing power of 2 based coefficients. Variable node updating occurs via (2.14) for all three algorithms.

2.3 Numerical Implementation

Minimum complexity implementation of the A-Min*-BP algorithm necessitates simulation of finite wordlength effects on edge metric storage (which dominates design complexity). Quantization selection consists of determining a total number of bits as well as the distribution of these bits between the integer and fractional parts *(I,F)* of the numerical representation. The primary objective is minimization of the total number of bits with the constraint that only a small performance degradation in the waterfall and error-floor BER regions is incurred. Quantization saturation levels ($Sat = 2^I$) that are too small cause the decoder to exhibit premature error-floor behavior. We have not analytically characterized the mechanism by which this occurs. However, the following provides a rule of thumb for the saturation level,

$$Sat = -\ln(p) \approx \ln\left(\frac{1-p}{p}\right) \qquad \text{where } p = e^{-Sat}.$$

This allows literal Log-Likelihood Ratio (LLR) representation of error probabilities that are as small as p. In practice, this rule seems to allow the error-floor to extend to a level that is about one order of magnitude lower than p.

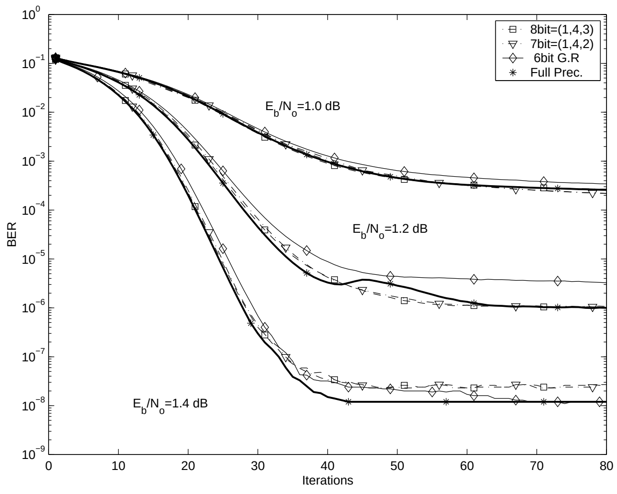
In the results that follow, simple uniform quantization has been employed, where the step size is given by 2^{-F}. To begin, Fig. 2.5 shows that low SNR performance is less sensitive to quantization than high SNR performance. A small but noticeable degradation occurs when 2 rather than 3 fractional bits are used to store edge metrics and 4 integer bits are used in both cases. In summary, 7 bits of precision (Sign, 4 Integer, 2 Fractional) are adequate for the representation of observation and edge metric storage in association with the considered code.

When power of 2 based quantization is used, the negative and positive saturation levels follow $[-2^{I-1}, 2^{I-1} - 2^{-F}]$. An alternative approach arbitrarily sets this range between a maximum and a minimum threshold and sets the step size equal to

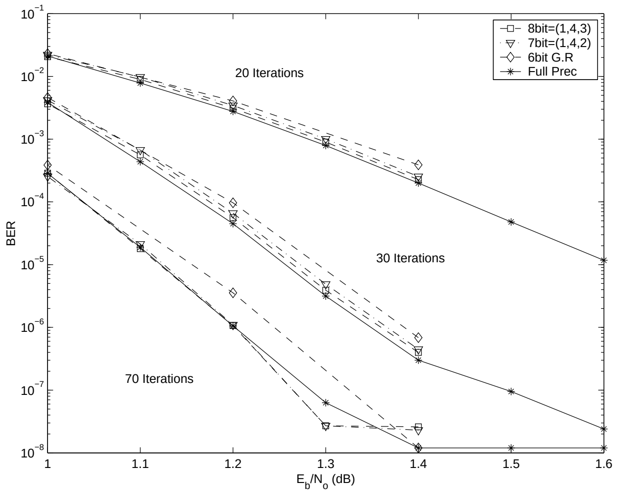
$s = 2 * MaxRange/2^{TotalBits}$. This approach to quantization is more general than the previous since the step size is not limited to powers of 2. We have found that in the low SNR regime, smaller quantization ranges are adequate, but the optimal step size remains similar to that needed at higher SNRs. Thus, operation at lower SNRs requires fewer overall bits given the general range approach to quantization. For example for $E_b/N_o = 1.0$dB, when $MaxRange = 10$ and a total of 6 bits were used, no performance degradation was observed. For higher SNR values, $MaxRange = 16$ was the best choice. This agrees with the results obtained using binary quantization with $(I, F) = (4, 2)$. The performance of this quantizer is described in Fig. 2.5 by the curve labeled '6bit G.R.' (or 6 bit general range) where in this case the range is set equal to (-10,10)@1.0dB;(-12,12)@1.2dB;(-16, 16)@1.4dB and a total of 6 bits (1 sign, 5 quant-bits) is used. Hence in this case the general range quantizer is equivalent to the (1,4,1) power of 2 quantizer at high SNR. At lower SNRs, the best case range was smaller than (-16,16) such that general range quantization offers an added degree of freedom in precision allocation that is useful in the context of LDPC decoding.

2.4 The UCLA LDPC Codec

We have implemented the above constraint update technique along with many other necessary functions in order to create a high throughput Monte Carlo simulation for arbitrary LDPC codes. The design runs on a VirtexII evaluation board from Nallatech systems and is interfaced to a PC via a JAVA API. A block diagram is provided in Fig. 2.6. The Gaussian noise generator developed by the authors in [13] is instantiated



(a)



(b)

Figure 2.5. (a) BER Vs. E_b/N_o, for different fixed numbers of iterations. (b) BER vs Iterations, for different fixed E_b/N_o.

next to the decoder so as to avoid a noise generation bottleneck. This block directly impacts the overall value of the system as a Monte Carlo simulator for error-floor testing as good noise quality at high SNR (tails of the Gaussian) is essential. Since the LDPC decoding process is iterative and the number of required iterations is non-deterministic,

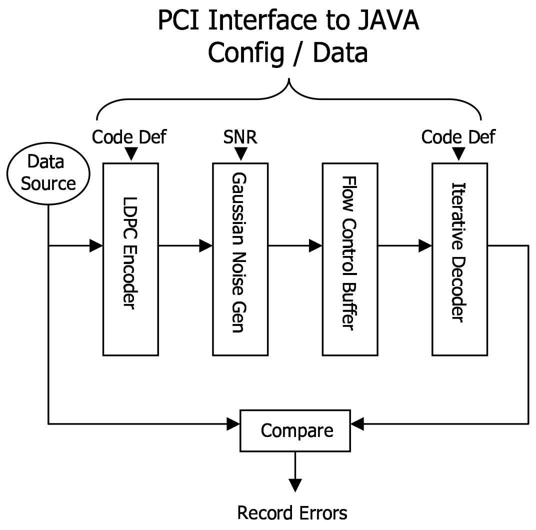


Figure 2.6. Architecture block diagram.

a flow control buffer can be used to greatly increase the throughput of the overall system.

Through the use of JAVA as an soft interface to the board, we have been able to facilitate the initiation and monitoring of simulations from remote locations. Researchers around the world have successfully uploaded their own LDPC codes for testing on the "UCLA Monte Carlo System".

2.5 Conclusion

A reduced complexity decoding algorithm that suffers little or no performance loss has been developed and is justified both theoretically and experimentally. Finite wordlengths have been carefully considered and 6 to 7 bits of precision have been shown to be adequate for a highly complex (a length 10,000 $d_{lmax} = 20$ irregular

LDPC) code to achieve an error floor that is code rather than implementation limited.

Chapter 3

Parity Check Matrix Construction for Error Floor Reduction

Density evolution determines the performance threshold for infinitely long codes whose associated bipartite graphs are assumed to follow a tree-like structure. However, bipartite graphs representing finite-length codes without singly connected nodes inevitably have cycles and thus are non-tree-like. Cycles in bipartite graphs compromise the optimality of the commonly practiced belief propagation decoding. If cycles exist, neighbors of a node are not conditionally independent in general, therefore graph separation is inaccurate and so is Pearl's polytree algorithm [34] (which defines belief propagation as a special case). However, not all cycles are equally problematic in practice. We will argue that the more connected a cycle is to the rest of the graph, the less difficulty it poses to iterative decoding.

Randomly realized finite-length irregular LDPC codes with block sizes on the order of 10^4 [36] approach their density evolution *threshold* closely (within 0.8dB at BER $\approx 10^{-6}$) at rate $1/2$, outperforming their regular counterparts [29] by about 0.6dB. In this work, we repeated the irregular code construction method described in [36] and extended their simulation to a higher SNR region. In the relatively unconditioned codes, an error floor was observed at BERs of slightly below 10^{-6}. In contrast, regular codes and almost regular codes ([24]) usually enjoy very low error floors, apparently due to their more uniform Hamming distance between neighboring codewords and higher minimum distances.

MacKay *et al.* [29] first reported the tradeoff between the threshold SNR and the error floor BER for irregular LDPC codes versus regular LDPC codes. A similar tradeoff has been found for turbo codes ([3], [16]). This work presents a design technique that requires all small cycles to have a minimum degree of connectivity with the rest of the graph. This technique lowers the error floors of irregular LDPC codes significantly while only slightly degrading the performance in the waterfall region.

The error floor of an LDPC code under maximum likelihood (ML) decoding depends on the d_{min} of the code and the multiplicity of d_{min} error events. However, for randomly constructed codes, no algorithm is known to check if they have large minimum distances (This problem was proved to be NP-hard [47]).

As a result, the common approach has been to indirectly improve d_{min} through code conditioning techniques such as the removal of short cycles (girth conditioning [31], [2]). Such conditioning is useful also because certain short cycles can cause poor performance in conjunction with iterative decoding even if they have a large d_{min} and would not be problematic for ML decoding.

However, not all short cycles are equally harmful. Standard girth conditioning severely constrains code structure by removing all cycles shorter than a specified length even though many of these can do little harm, because they are well-connected with the rest of the graph. This work uses a technique precludes only cycles that are relatively isolated from the rest of the graph and thereby are likely contributors to small stopping sets. Di *et al.* [12] described stopping sets in the context of erasures. Message passing decoding fails whenever all the variable nodes in a stopping set are erased. Notably, erasing all variable nodes in a stopping set does not necessarily force ML decoding to fail.

For QPSK and BPSK modulation, Euclidean distance and Hamming distance are linearly related. Thus, it is reasonable to design codes for such modulations to focus on the Hamming distance spectrum. Minimum Hamming distance (d_{min}) is well-known to be related to the number of errors (t) that can be corrected reliably, $d_{min} = 2t + 1$. Minimum Hamming distance is also known to be linearly related to the number of erasures (u) that can be corrected, $d_{min} = u + 1$. As we will show, the minimum stopping set size is equal to the smallest number of erasures that cannot be corrected by iterative decoding. Thus it is closely related to the Hamming distance (and hence Euclidean distance for QPSK and BPSK). Because the weight distribution of stopping sets is so closely related to the Euclidean distance spectrum, it is clear that LDPC designs focusing on the weight distribution of stopping sets is appropriate for AWGN channels as well as the binary erasure channel (BEC).

The focus of this work is on LDPC codes for AWGN channels, but we improve erasure performance (stopping set size) to indirectly improve AWGN performance. As demonstrated by our simulations, we can reduce error floors in AWGN channels by generating codes from an expurgated ensemble that has large stopping sets. The resulting codes outperform the girth-conditioned irregular codes described in [31] and [2]. The work is organized as follows. Section 3.1 shows the relationship between cycles, stopping sets, and codewords. Section 3.2 proposes a design metric termed *extrinsic message degree* (EMD). Based on the approximate cycle EMD (ACE), section 3.3 describes a linear-time Viterbi-like algorithm to construct full-rank parity-check matrices that follow irregular degree distributions but do not have isolated small cycles. This indirectly forces a large stopping set size. Section 3.4 demonstrates by simulation the substantial decrease in error floor achieved by ACE-conditioned codes as compared to girth-conditioned codes.

3.1 Cycles, stopping sets, codeword sets and edge-expanding sets

The well known matrix and bipartite graph descriptions of a rate 1/3 $(9, 3)$ code are given in Fig. 3.1. This code will be used in examples throughout the chapter. One column in the parity-check matrix corresponds to one variable in the bipartite graph. For convenience, we will use ‘column’ and ‘variable’ interchangeably in this chapter. The parity-check matrix H is constructed such that its H_2 portion is invertible, which guarantees that H is full-rank. For systematic encoding, H_1 corresponds to information bits and H_2 corresponds to parity bits.

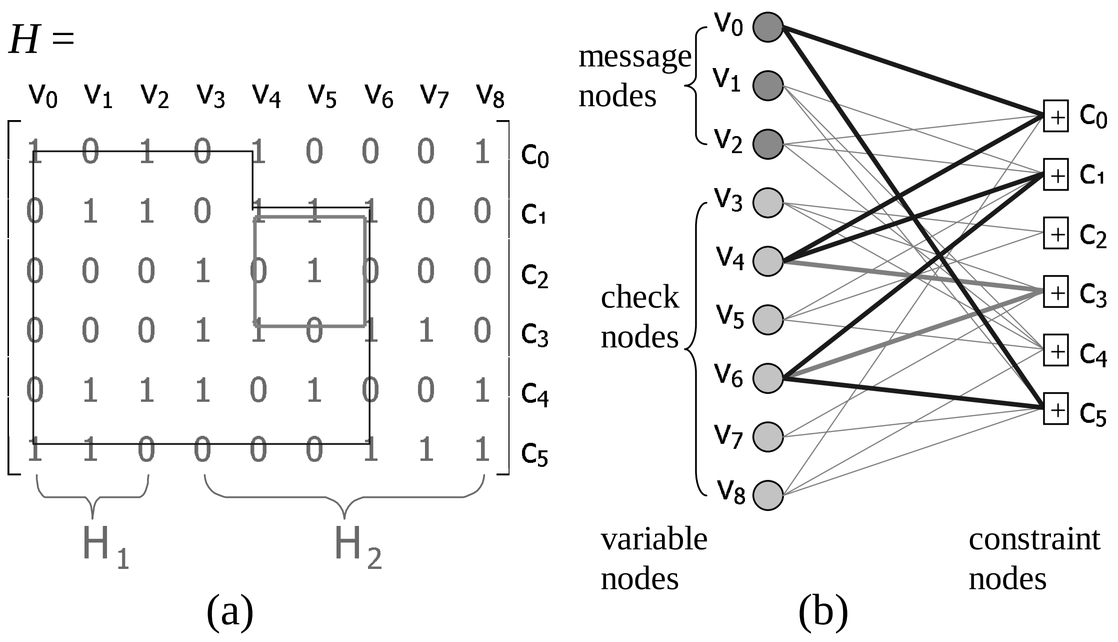


Figure 3.1. Matrix and graph description of a (9, 3) code.

3.1.1 Cycle-related structures

Definition 1 *(**Cycle**) A cycle of length $2d$ is a set of d variable nodes and d constraint nodes connected by edges such that a path exists that travels through every node in the set and connects each node to itself without traversing an edge twice.*

Definition 2 *(C_d **Cycle set**) A set of variable nodes in a bipartite graph is a C_d set if (1) it has d elements, and (2) one or more cycles are formed between this set and its neighboring constraint set. A set of d variable nodes does not form a C_d set only if no cycles exist between these variables and their constraint neighbors.*

Note that the maximum cycle length that is possible in a C_d set is $2d$. Fig. 3.1 shows a length-6 cycle ($v_0 - c_0 - v_4 - c_1 - v_6 - c_5 - v_0$) and a length-4 cycle ($v_4 - c_1 - v_6 - c_3 - v_4$). Variable node set $\{v_0, v_4, v_6\}$ is a C_3 set. Variable node set $\{v_4, v_5, v_6\}$ is also a C_3 set although v_5 is not contained in the length-4 cycle. Di *et al.* defined a stopping set as follows, which we will show to contain cycles shortly.

Definition 3 *(*S_d **Stopping set [12]***) A variable node set is called an* S_d *set if it has* d *elements and all its neighbors are connected to it at least twice.*

Variable node set $\{v_0, v_4, v_6\}$ in Fig. 3.1 is an S_3 set because all its neighbors c_0, c_1, c_3 and c_5 are connected to this set at least twice.

The following lemma shows that stopping sets always contain cycles. The effectiveness of message passing decoding on graphs with cycles depends primarily on how cycles are clustered to form stopping sets.

Lemma 1 *In a bipartite graph without singly connected variable nodes (such as one generated with a degree distribution given by density evolution), every stopping set contains cycles.*

Proof: A stopping set (variable nodes) and its neighbors (constraint nodes) form a bipartite graph where one can always leave a node on a different edge than used to enter that node. Traversing the resulting bipartite graph in this way indefinitely, one eventually visits a node twice, thus forming a cycle.□

Lemma 2 *In a bipartite graph without singly connected variable nodes, stopping sets in general are comprised of multiple cycles. The only stopping sets formed by a single cycle are those that consist of all degree-2 variable nodes.*

Proof: A cycle that consists of all degree-2 variable nodes is a stopping set. To prove the lemma, we only need to show that if a cycle contains variable nodes of degree-3 or more, any stopping sets including this cycle are comprised of multiple cycles. Fig. 3.2(a) shows a cycle of arbitrary length $2d$ (here $2d = 8$ for demonstration). Assume that one variable node v_2 in this cycle has degree 3 or higher, v_2 must be connected to at least one constraint node out of this cycle (for instance c_1 in Fig. 3.2(a)). By the definition of a stopping set, c_1 must be connected to variable nodes in the stopping set at least twice. Therefore if c_1 is not connected to v_1, or v_3, or v_4, the stopping set must contain at least one more variable node (for instance v_5). The 'concatenation' of constraints and variables on to v_5 may occur across many nodes. However, to form a stopping set, eventually a new loop must be closed that connects the newest constraint in the chain to a variable on the chain or in the original cycle. Thus, the stopping set is comprised of at least two cycles.$\square$

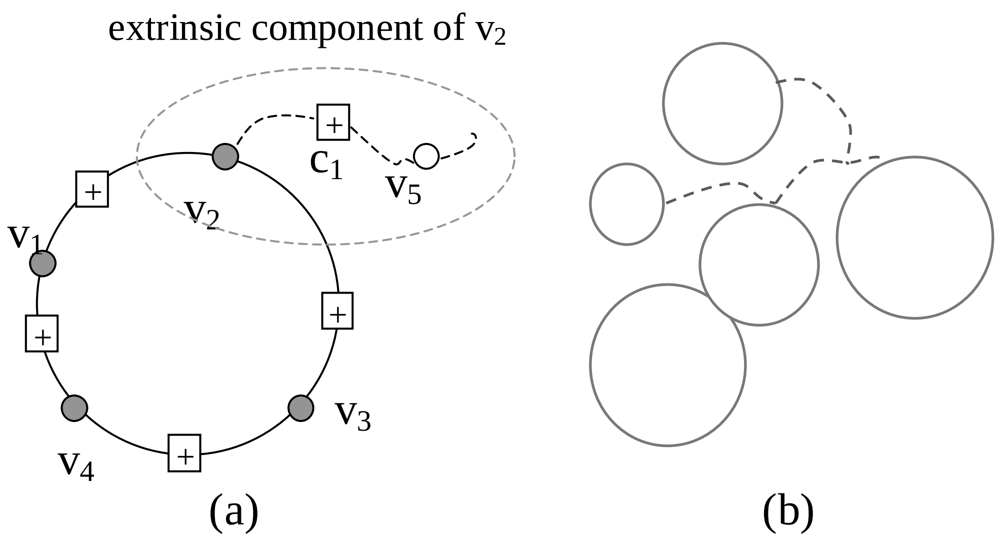


Figure 3.2. (a) Extrinsic message (b) Expanding of a graph

According to Lemma 2, the general view of stopping sets and cycles is given in Fig. 3.2(b). Two types of variable nodes comprise a stopping set. Variable nodes of the first type form cycles with other variable nodes; variable nodes of the second type form binding structures that connect different cycles. It should be noted that both binding nodes and cycle nodes may have branches that lead to cycles containing variable nodes not in the current stopping set. Our proposed parity matrix design algorithm ensure that short cycles contain at least a given minimum number of 'extrinsic paths'. This leads to an increase in the minimum size of a stopping set.

Definition 4 *(*W_d **Codeword set***) A variable node set is called a* W_d *set if it is comprised of exactly* d *elements whose columns form a (weight-d) codeword.*

Variable nodes set $\{v_0, v_4, v_6\}$ in Fig. 3.1 is the W_3 set corresponding to the codeword 100010100. A linear code with minimum distance d_{min} has at least one codeword with weight d_{min} and no non-zero codewords with smaller weight. Hence, there is at least one $W_{d_{min}}$ set but no W_d sets where $d < d_{min}$.

Erasing all the variables in a codeword set is the same as erasing all the non-zero positions of a binary codeword. Recovery from such an erasure is impossible even under ML decoding. Thus all codeword sets are stopping sets.

Preventing small stopping sets also prevents small d_{min}. If a code has d_{min}, it must have an $S_{d_{min}}$ stopping set. Thus, avoiding all stopping sets S_d for $d \leq t$ ensures $d_{min} > t$.

However, small stopping sets do not necessarily represent low distance events. Indeed, an ML decoder can successfully decode an erased stopping set if a full column-rank sub-matrix is formed by the columns of the parity check matrix that are indexed by the stopping set variables. For example, $\{v_3, v_4, v_5, v_6, v_8\}$ in Fig. 3.1 is a stopping set that may be recovered by ML decoding (in the BEC case, simply solve a linear equation set). However, an erased stopping set can never be overcome by an iterative decoder.

With additive white Gaussian noise (AWGN), the magnitude of a corrupted signal can be so small that it can be effectively treated as an erasure. Hence the role of stopping sets can be translated to AWGN scenarios where variables with poor observation reliability are analogous to erasures. All stopping sets of small size are problematic. Some cause small distance, and all cause problems for iterative decoding. An obvious direction to take in order to generate codes well-suited to iterative decoding is to increase the size of minimum stopping set and reduce its multiplicity. Fig. 3.3 summarizes the relationship between C_d, S_d, and W_d. Here, E_d is a graph structure that we will discuss in the next section.

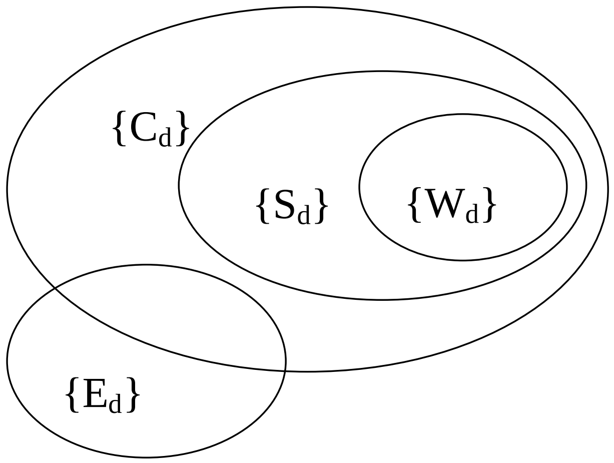


Figure 3.3. Venn diagram showing relationship of C_d, S_d, W_d and E_d.

3.1.2 Cycle-free sets

At this point, the value of removing small stopping sets is apparent. However, one might argue that simple girth conditioning accomplishes this because every stopping set contains cycles. The problem with traditional girth conditioning is that there are so many cycles. Fig. 3.4 illustrates a cycle in the support tree of variable node v_0 of Fig. 3.1. All the levels whose indices are odd numbers consist of constraint nodes and all the levels whose indices are even numbers consist of variable nodes. A cycle occurs if two positions in the support tree represent the same node in the bipartite graph (*e.g.*, v_8 in level-3). To detect cycles of length up to $2d$ in the support tree of v_0, we need to expand its support tree d levels.

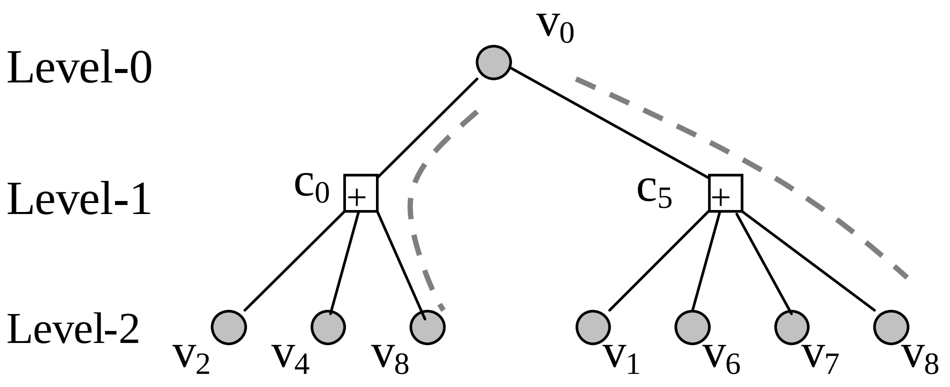


Figure 3.4. Traditional girth conditioning removes too many cycles.

The number of nodes in the support tree grows exponentially with the number of levels expanded. To be short-cycle-free, all these nodes have to be different, so the longest cycle size we can avoid increases only logarithmically with block size (see [18]). Since the logarithm is a slowly increasing function, girth conditioning of a finite length LDPC code is severely limited by block length.

Girth conditioning is especially problematic when there are high-degree nodes, as is common with degree distributions produced by density evolution. Recent girth conditioning techniques usually bypass high degree nodes. For example, in [31], the edge-wise highest variable degree is only 3; in [2], the fraction of the highest degree variables λ_8 is only 0.025. As a result, girth conditioning was easier to perform. However, the capacity-approaching capability was sacrificed. High degree nodes are indicated by density evolution and lead to large stopping sets. The following arguments further discuss the cycle-related structures for high degree nodes and low degree nodes.

Definition 5 *(***Cycle-free set***) A variable node set is called a cycle-free set if no cycle exists among its constituent variables.*

Theorem 1 *A necessary and sufficient condition for a set of degree-2 variable nodes to be a cycle-free set is that this set is linearly independent.*

Proof: All sets that are not linearly independent contain codeword sets. Codeword sets are special stopping sets and stopping sets contain cycles (Lemma 1). For sufficiency, note that the constraint nodes taking part in a cycle among degree-2 nodes are each shared by exactly two variable nodes. Therefore the binary sum of columns (variables) taking place in the cycle is the all-zero vector and these columns are linearly dependent.□

Corollary 1.1 *A maximum of $n - k - 1$ degree-2 columns of length $n - k$ may be linearly independent (cycle-free).*

Proof: Consider the $(n - k) \times (n - k - 1)$ bi-diagonal matrix,

$$\begin{array}{cc} \begin{bmatrix} 1 & 0 & \cdots & & 0 \\ 1 & 1 & & & \\ 0 & 1 & \ddots & & \\ \vdots & & \ddots & & \\ & & & 1 & 0 \\ & & & 1 & 1 \\ 0 & \cdots & & 0 & 1 \end{bmatrix} & {\scriptstyle n-k}, \\ {\scriptstyle n-k-1} & \end{array} \tag{3.1}$$

This matrix forms a rank $n - k - 1$ basis of degree-2 columns each with dimension $n - k$. Any possible degree-2 column of dimension $n - k$ can be formed via a linear combination of columns in the above basis.□

Corollary 1.1 may also be considered a version of the Singleton bound where the restriction to degree-2 columns lowers the best possible d_{min} from $n - k$ to $n - k - 1$.

Theorem 2 *In an* (n, k) *code free of degree-1 variables, a cycle-free variable node set* $v_1, v_2, ..., v_s$ *must satisfy* $\sum_{i=1}^{s} (d_i - 1) \leq n - k - 1$, *where* d_i *is the degree of* v_i.

Proof: A degree-d variable node whose constraints are $\{c_1, c_2, ..., c_d\}$ can be conceptually replaced by a cluster of $d - 1$ degree-2 nodes whose constraints are $\{c_1, c_2\}$, $\{c_2, c_3\}$, ..., $\{c_{d-1}, c_d\}$ respectively. As an example, Fig. 3.5 shows how variable node v_1 in Fig. 3.1 may be replaced by a cluster of two degree-2 nodes.

Because the indices of the constraints in a cluster are ordered, any cycle involving the degree-d node is equivalent to a cycle involving some of the degree-2 nodes in the cluster replacing the original node. Fig. 3.6 shows an example. Replace every variable

$$\text{variable node } \mathrm{v}_1\text{:} \quad \begin{pmatrix}0\\1\\0\\0\\1\\1\end{pmatrix}\begin{matrix}\ldots c_0\\ \ldots c_1\\ \ldots c_2\\ \ldots c_3\\ \ldots c_4\\ \ldots c_5\end{matrix} \qquad \text{cluster of degree-2 nodes:} \quad \underset{(\mathrm{c}_1,\mathrm{c}_4)}{\begin{pmatrix}0\\1\\0\\0\\1\\0\end{pmatrix}} \quad \underset{(\mathrm{c}_4,\mathrm{c}_5)}{\begin{pmatrix}0\\0\\0\\0\\1\\1\end{pmatrix}}$$

Figure 3.5. v_1 **can be replaced by two degree-2 nodes.**

in the set with its equivalent cluster. According to Corollary 1.1, at most $n - k - 1$ of these resulting degree-2 nodes can form a cycle-free set. Thus the theorem is proved.□

Corollary 2.1 *In an (n, k) code free of degree-1 variables, no set of variable nodes whose cardinality is larger than $n - k - 1$ can be cycle-free.*

Proof: Equality in the inequality of Theorem 2 is achieved with $n - k - 1$ independent degree-2 variable nodes. Fewer variable nodes are allowed if some have higher degree.□

Lemma 2 and Theorem 1 show that for degree-2 variable nodes, cycles, stopping sets and codeword sets are equivalent. These structures are distinct for higher degree nodes following the Venn diagram of Fig. 3.3. Theorem 2 shows that higher degree nodes may be considered as the superposition of two or more degree-2 nodes, and they tend to form cycles more easily. As a result, girth conditioning on high degree nodes is more difficult than on low degree nodes. However, cycles involving higher degree nodes are actually less problematic because they have higher EMD, as will be explained in the next section.

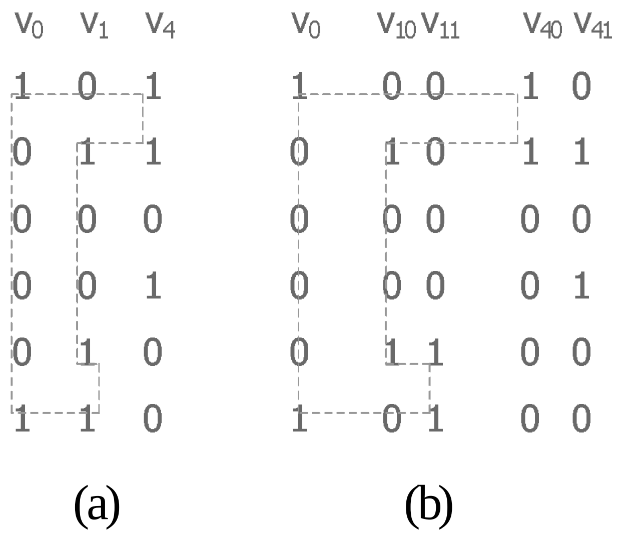


Figure 3.6. Replace v_1 by its cluster in a cycle.

3.2 EMD and LDPC code design

Our goal is to ensure that all stopping sets have at least some minimum number of variable nodes. However, no polynomial-time algorithm is known that removes small stopping sets explicitly, and our attempts to directly control stopping set sizes were too complex even to prevent very small stopping sets in a reasonable amount of time. As an alternative, we design an algorithm that conditions the graph so that small cycles all have an edge expanding property that prevents them from being part of a small stopping set. First, we define the notion of an edge expander as presented in [12].

Definition 6 *(**(α, γ) edge expander** [12]) Let L be any subset of left nodes (variable nodes). Define $E(L)$ to be the number of edges connected to L and $N(L)$ to be the number of neighbors of L. An (α, γ) edge expander of an (n, k) code is a graph that has $N(L) > \gamma E(L)$ for all subsets with $E(L) \leq \alpha n$.*

Methods that realize regular graphs with good edge expanding properties were proposed by Margulis in [32]. However, a construction that can simultaneously satisfy a given edge expanding property as well as a given irregular degree distribution has yet to be proposed. We are interested in the special case of $(\alpha, 1/2)$ edge expanders.

Definition 7 *(E_d* **Edge-expanding set with parameter** $\gamma = 1/2$*) A set of d variable nodes is called an E_d set if one-half of the number of edges emanating from it is less than the number of neighbors to which these edges connect.*

Neighbors (constraints) of an E_d set are connected to the set less than twice on average, therefore at least one neighbor exists that is singly connected to this E_d set. Thus, as shown in Fig. 3.3, the set of S_d sets and the set of E_d sets are disjoint. For a given value of d, increasing the number of E_d sets decreases the number of possible S_d sets. Hence, conditioning for good edge expansion indirectly reduces the number of small stopping sets.

Lemma 2 shows that stopping sets are comprised of linked cycles. An efficient way to suppress small stopping sets is to improve the edge expanding properties of cycles in an irregular LDPC code. From the discussion of E_d sets, we know that constraint nodes singly connected to a variable node set provide good edge expansion because these constraint nodes ensure useful message flows. Our algorithm achieves this by focusing on a parameter of variable node sets that we call the extrinsic message degree (EMD).

Definition 8 *(***Extrinsic message degree***) An extrinsic constraint node of a variable node set is a constraint node that is singly connected to this set. The extrinsic message* *degree (EMD) of a variable node set is the number of extrinsic constraint nodes of this variable node set.*

As was previously described, the high-SNR performance of an iteratively decoded LDPC code is limited by the size of the smallest stopping set in the code. The EMD of a stopping set is zero. A set of variable nodes with large EMD will require additional concatenation of nodes to become a stopping set. We propose a conditioning algorithm that ensures all cycles less than a given length have an EMD greater than a given value. This technique statistically increases the smallest stopping set size. It also increases d_{min} because codeword sets are special cases of stopping sets.

The algorithm that follows ignores large cycles. In the context of EMD, this is justified for two reasons. First, large cycles necessarily contain many variable nodes. Second, they tend toward high EMD by virtue of a level of graph connectivity that statistically surpasses that of small cycles. We thus target the elimination of small cycles with low EMD and claim (via the arguments of previous sections) that these structures are significant contributors to errors at high SNR.

3.3 Construction of LDPC codes free of small stopping sets

First consider the EMD of a generic cycle. If there are no variable nodes in a cycle that share common constraint nodes outside of the cycle, then the EMD of this cycle is $\sum_i (d_i - 2)$, where d_i is the degree of the i^{th} variable in this cycle. Otherwise, the

EMD is reduced through constraint node sharing. To provide a calculable EMD metric, we neglect constraint node sharing and define an approximate cycle EMD.

Definition 9 *(***Approximate cycle EMD (ACE)***) The ACE of a length $2d$ cycle is $\sum_i (d_i - 2)$, where d_i is the degree of the i^{th} variable in this cycle. We also say that the ACE of a degree-d variable node is $d - 2$ and the ACE of any constraint node is 0.*

ACE is an upper bound on EMD. The code conditioning algorithm to be proposed next is based on ACE instead of EMD. This approximation is reasonable since in this algorithm, *all* cycles shorter than a given length (including those formed through constraint node sharing) will be required to meet the ACE criteria. An LDPC code has property (d_{ACE}, η_{ACE}), if all the cycles whose length is $2d_{ACE}$ or less have ACE values of at least η_{ACE}.

In our codes information bits come before parity bits (see Fig. 3.1). We assign column nodes such that degree decreases monotonically (*i.e.*, $d_i \geq d_j$ if $i < j$). Because high degree nodes converge faster, this arrangement provides more protection to information bits than to parity bits. The algorithm is as follows:

```
for ($i = n - 1; i \geq 0; i - -$)
begin
redo:
    Randomly generate $v_i$ according to deg. distr.;
    if $i \geq k$ (i.e., $v_i$ is a parity bit)
    begin
        Gaussian Elimination (GE) on $H_2$;
        if $v_i \in SPAN(v'_{i+1}, v'_{i+2}, ..., v'_{n-1})$
            goto redo;
        else
            $v'_i \leftarrow$ the residue of $v_i$ after GE;
    end
    ACE detection for $v_i$;
    if $ACE < \eta_{ACE}$ for a cycle of length $2d_{ACE}$ or less
        goto redo;
end
```

The Gaussian elimination process ultimately guarantees that the H matrix has full rank by ensuring that the $n-k$ columns of H_2 will be linearly independent. For degree-2 variable nodes, independence entails freedom from cycles so that all degree-2 parity check nodes will be cycle-free. A caveat is that if Gaussian elimination is used in conjunction with a degree distribution that yields more than $n-k$ degree-2 nodes, then at least one of the $n-k$ parity check variables should have odd number degree (this can be achieved by column swapping). This follows immediately from Corollary 1.1.

The ACE detection method can be equivalently depicted in two ways. The first one, based on support trees, is directly related to the graph structure. The second one, based on trellises, is oriented for algorithm implementation.

The tree depiction of ACE detection ($\eta_{ACE} = 0$) is given in Fig. 3.7. Here, variable and constraint node labels refer literally to those of the example code in Fig. 3.1 and the support tree that extends four levels below root node v_0 is portrayed. We define p_t to be the ACE of a path between root node v_0 and an arbitrary node μ_t (it can be either a variable node or a constraint node). Recall also that $ACE(\mu_t) = degree(\mu_t) - 2$ if μ_t is a variable, and $ACE(\mu_t) = 0$ if μ_t is a constraint.

ACE Detection of v_0

```
$p_t \leftarrow \infty$ for all variables and constraints;
$p_0 \leftarrow ACE(v_0)$; Activate $v_0$ for level-0;
**for** ($l = 1$; $l \leq d_{ACE}$; $l++$)
**begin**
    **for** any active node $w_s$ in level-($l - 1$)
    **begin**
        Find its children set $Ch(w_s)$;
        **for** every child $\mu_t \in Ch(w_s)$
        **begin**
            $p_{temp} \leftarrow p(w_s) + ACE(\mu_t)$;
            **if** $(p_{temp} + p_t - ACE(v_0) - ACE(\mu_t)) < \eta_{ACE}$[1]
                **exit** with failure;
            **elseif** $p_{temp} \geq p_t$
                Deactivate $\mu_t$ in level-$l$;
            **else**
                $p_t \leftarrow p_{temp}$;
        **end**
    **end**
**end**
**exit** with success;
```

[1]Note that if $(p_{temp} + p_t - ACE(v_0) - ACE(\mu_t)) < \infty$, this is the ACE of a cycle involving μ_t.

To explain the above algorithm, we need to recognize that a node should propagate descendants (be active) only if the path leading to this node has the lowest ACE value that any path to this node has had thus far. Therefore linear cost is achieved instead of an exponential cost. Initially all the path ACEs can be set to ∞ (which means 'unvisited'). Note that cycles occur when a node is revisited, is simultaneously visited, or both. A *cycle ACE* equals the sum of the previously lowest path ACE to a node and the current path ACE to the node minus the doubly counted root and child ACE. When a cycle is formed by connecting two distinct paths from v_0 to μ_t we have cycle $ACE = p_{temp} + p_t - ACE(v_0) - ACE(\mu_t)$, where p_{temp} and p_t are the ACEs of the two paths from v_0 to μ_t. Handling multiple simultaneous arrivals to the same node is a trivial extension where ACE minimization is performed sequentially across all arrivals.

In the example shown in Fig. 3.7, bold lines at each level describe the current set of *active paths*. In this example 'ties' are assigned the path whose parent has the lowest index. For instance the path (v_0-c_5-v_1-c_1) with $ACE = 1$ survives while, (v_0-c_5-v_6-c_1), (v_0-c_0-v_2-c_1), (v_0-c_0-v_4-c_1) each also having $ACE = 1$, perish. For an example of pruning occurring due to cycle detection on differing levels of the tree, observe that the path (v_0-c_0-v_8-c_5) with $ACE = 1$ does not survive since c_5 was visited at Level-1 and was accordingly assigned $ACE = 0$.

Fig. 3.8 provides a trellis depiction of the previous discussion (with two more stages added). A trellis instead of a full support tree is adequate for ACE detection because the ACE minimization is performed sequentially and only the minimum ACE needs to be stored. Again, a path ACE is stored for every variable node and every constraint node. An active path is a path that connects the root variable node and any other node with

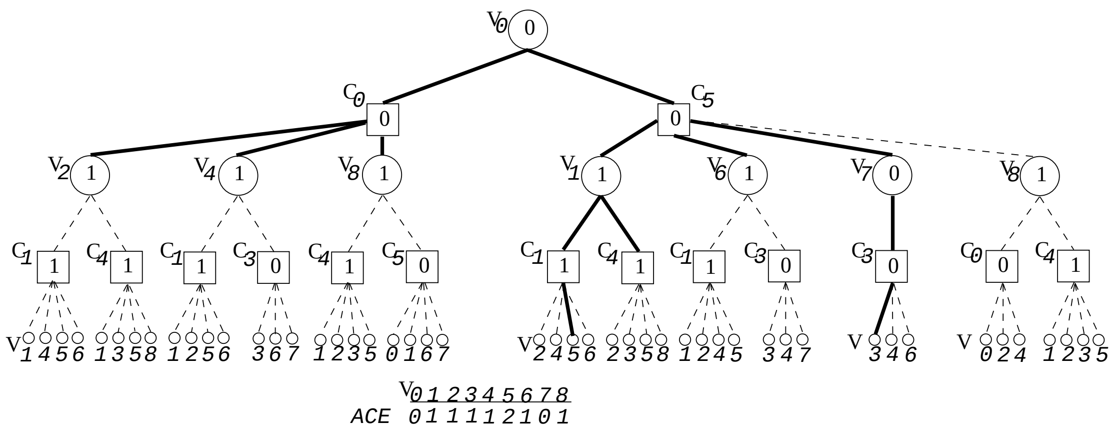


Figure 3.7. **Illustration of an ACE search tree associated with v_0 in the example code of Fig.3.1. $\eta_{ACE} = 0$. Bold lines represent survivor paths. ACE values are indicated on the interior of circles (variables) or squares (constraints), except on the lowest level where they are instead described with a table.**

the lowest ACE value up to the current step. Active paths are marked by solid lines in Fig. 3.8. An *active node* is a node that connects to an active path at the current step.

Viterbi tree pruning yields a complexity at each root that is upper bounded by $d_{ACE} \times n \times (d-1)$ where d is the highest degree of any node in the graph, because the support tree is expanded d_{ACE} levels, for each level we have to consider at most n nodes, and every node has at most $d-1$ children. As shown in Fig. 3.8, relatively few nodes at a level are active, thus the actual computational burden is reasonable, even for block size on the order of 10^5 bits. The storage space needed is on the order of $n + (n-k)$ because only the current trellis level has to be saved. To further improve run-time, any active node with path ACE $p_t \geq \eta_{ACE}$ can be deactivated because all the paths stemming from this node have an ACE value of at least η_{ACE}. As a special

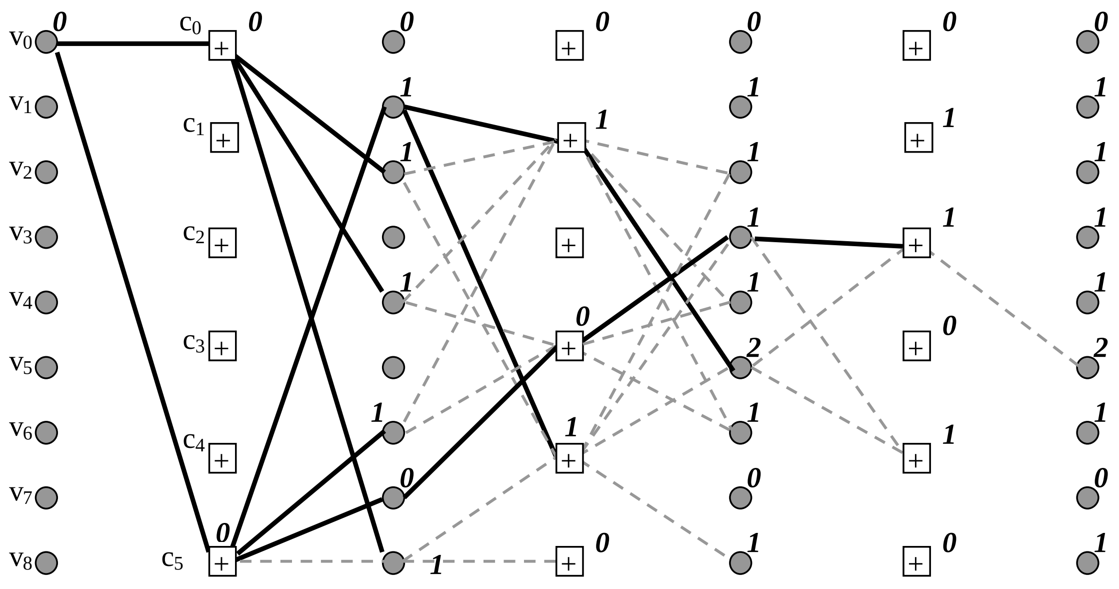


Figure 3.8. The Viterbi-like ACE algorithm. $\eta_{ACE} = 0$

application of this argument, the user may generate columns for all variable nodes with degree $\eta_{ACE} + 2$ or higher without checking the ACE (since all descendants will have a path ACE of at least η_{ACE}).

Note that cycle detection can be implicitly performed in the above algorithm. Unvisited variables are initialized to $p_t = \infty$. When a variable is first visited, p_t is assigned a finite value. Upon subsequent visits to the same variable, ill-conditioned cycle detection is alarmed if the condition $(p_{temp} + p_t - ACE(v_0) - ACE(\mu_t)) \geq \eta_{ACE}$ is not satisfied.

3.4 Simulation results and data analysis

We present results for LDPC codes with three different block lengths.

3.4.1 Block-length 10,000 LDPC codes

We used the ACE algorithm to construct (10000, 5000) codes that have the irregular degree distribution given in [36] with maximum variable node degree $d_v = 20$. The encoded bits were sent through a binary-input AWGN channel. For each corrupted codeword, a maximum of 200 iterations were performed. Each simulation was stopped when 80 block errors were detected. The BER results and in one case the word (block) error rate (WER) results are plotted in Fig. 3.9.

A (d_{ACE}, η_{ACE}) code in Fig. 3.9 means that in this code, all cycles of length up to $2d_{ACE}$ have ACE of at least η_{ACE}. Higher η_{ACE} values ensure better properties for the conditioned cycles and higher d_{ACE} values ensure that conditioning occurs for longer cycles. There is a tradeoff between d_{ACE} and η_{ACE}: increasing one parameter inevitably makes the other parameter more difficult to implement. For $d_{ACE} = 13, 10, 9, 7$, and 6, the highest η_{ACE} values that we could achieve were $2, 3, 4, 5$, and 7 respectively. If all degree-2 variables are parity bits (which is possible if there are fewer than $n-k$ of them), then the full-rankness of the parity matrix guarantees that no cycles will exist between degree-2 variables (see Theorem 1). In this case we can construct (∞, 1) codes since all cycles of any length must include nodes of at least degree-3 (and hence have ACE of at least 1). Such codes have the lowest level of conditioning that we consider and we use codes constructed in this manner to contrast the performance of stronger (d_{ACE}, η_{ACE}) conditioning levels.

As a benchmark, Richardson and Urbanke's 10^4-bit code [36] (referred to here as the RU code) is included in Fig. 3.9(a). This RU code was constructed with the constraint that “all the degree-two nodes were made loop-free”. Unfortunately, simulation results below BER 10^{-6} are unavailable for the RU code. The (∞, 1) code shown in this chapter has a similar level of conditioning as the RU code and exhibit comparable performance. The (∞, 1) code ensures linear independence among parity bits which include all degree-two nodes and some degree-three nodes. The error-floor BER of the (∞, 1) code is around 10^{-6}. Pure length-4 cycle removal improves BER by half an order in magnitude over the (∞, 1) code at highest SNR, and adversely affects convergence. However, proper selection of d_{ACE} and η_{ACE} suppresses error floors significantly more. For example, the pure ACE conditioning code (9, 4) achieves approximately BER = 10^{-9}, three orders of magnitude below the (∞, 1) code.

Fig. 3.9(b) compares several ACE parameter sets with or without explicit length-4 cycle removal. Note that the lowest error floor was achieved at $d_{ACE} = 9$ and $\eta_{ACE} = 4$ with no explicit removal of short cycles. As explained before, traditional girth conditioning treats all short cycles equally, thus making the removal of longer but still harmful cycles more difficult. On the contrary, the ACE algorithm effectively removes low-ACE cycles, while leaving shorter cycles intact, if they have a high ACE. By doing this, participation of high degree variables in cycles is encouraged. In fact, removing all length-4 cycles hinders the performance of the ACE algorithm.

We observe that there is a small penalty in the capacity-approaching capability of our low-error-floor codes. With more conditioning at this block size, the low-SNR performance of the code is slightly degraded, possibly due to a decrease in the randomness of the code structure. This tradeoff between error floor and low-SNR performance is a well-known characteristic of iteratively decoded codes ([29][3][16]). Scheme (9,

4) is approximately 0.07dB away from the RU code at low SNR. However, even with this mild penalty these codes remain superior to regular codes in terms of their capacity-approaching performance. For example, we tested MacKay's (9972, 3, 6) regular LDPC code described in [29]. Although no error floors were detected for this code, it achieves $\mathrm{BER} \approx 10^{-5}$ at $SNR \approx 1.7$dB, more than 0.6dB worse than our (9, 4) code. Thus the combination of density evolution optimized degree distributions and ACE construction achieves good performance over a wide SNR operating range.

3.4.2 Shorter block lengths

To compare with other techniques at block lengths around 1000, we choose Mao's (1268, 456) code described in [31] as a benchmark. Fig. 3.10 compares the performance of the ACE-conditioned (1264, 456) code with that of Mao's (1268, 456) code (results drawn from [31]).

It should be noted that degree distributions play an important role in conditioning schemes. With the low maximum variable degree ($d_v = 3$) distribution proposed in [31], both the girth conditioning and the ACE conditioning are easy to perform. However, the ACE-conditioned code (∞, 3) is 0.2 dB better in BER at the high SNR region. If the density-evolution optimized distribution with $d_v = 14$ is imposed, the girth conditioning technique becomes more difficult due to the higher fraction of high degree nodes. However, the ACE algorithm still works well, outperforming Mao's code by 0.3dB, with no detectable error floors above $BER = 10^{-9}$.

We have also designed two ACE-conditioned (4000, 2000) codes to compare with

Arnold's (4000, 2000)($d_v = 8$) code described in [2] (results drawn from [2], see Fig. 3.11). The degree distributions of the proposed code are Arnold's $d_v = 8$ degree distribution and the $d_v = 15$ degree distribution produced by density evolution.

As we can see from Fig. 3.11, by choosing Arnold's degree distribution, the ACE-conditioned code (∞, 6)[2] achieves convergence 0.4dB worse than Arnold's code. This may result from the fact that the ACE algorithm is based on random generation and Arnold's progressive edge-growth technique successfully puts more structure in codes. We did not find error floors for the (∞, 6) code above BER $= 10^{-8}$. By choosing the density-evolution optimized distribution with $d_v = 15$, the ACE-conditioned code (9, 4) achieves a threshold SNR 0.1dB better than that of Arnold's code.

In summary, Arnold's code is better than the ACE-conditioned code for Arnold's $d_v = 8$ degree distribution, but that degree distribution is suboptimal. ACE conditioning of the density evolution degree distribution produces the best performance.

3.5 Conclusion

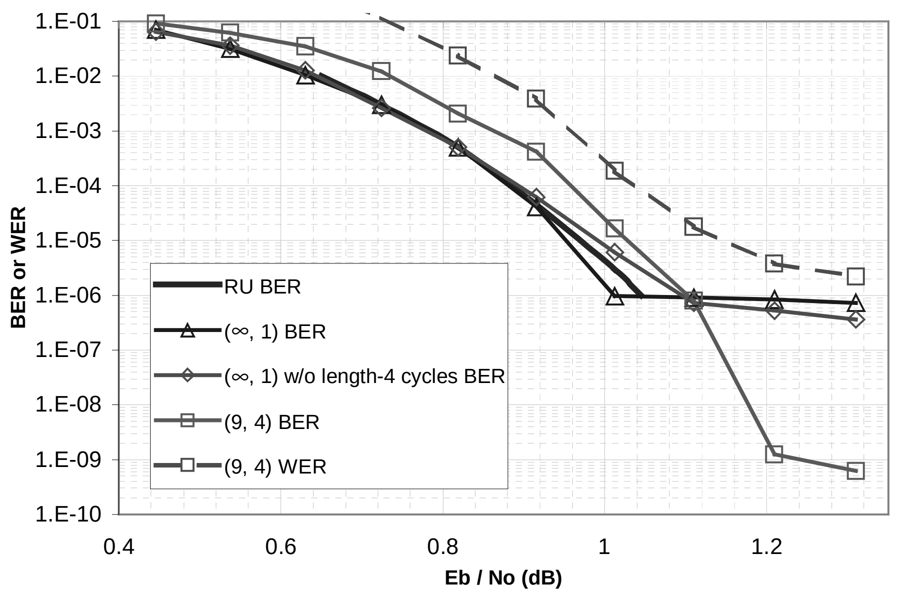
We have discussed the relationship between several graph structures that affect error floors and have introduced the ACE algorithm to construct irregular LDPC codes that have a specific ACE property for short cycles while maintaining the density-evolution prescribed degree distribution. Our simulation results show an improvement in the error floor region of irregular LDPC codes by several orders of magnitude in BER over alternative construction techniques.

[2] $d_{ACE} = \infty$ is achieved by extending the trellis until all the notes in the rightmost level are inactive

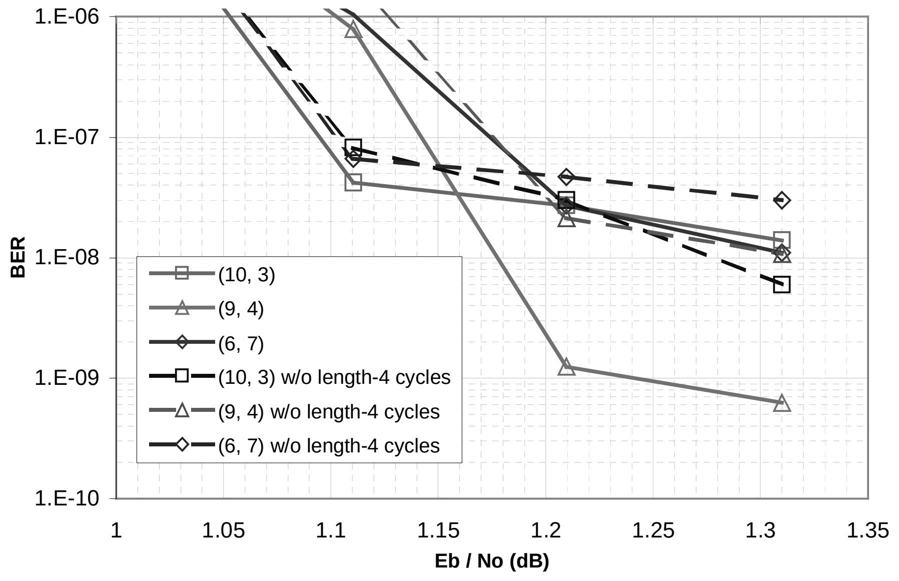
Graphs with lower maximum left degree can achieve higher girth conditioning numbers than graphs with higher maximum left degree. Following this mechanism, [31] and [2] demonstrate that error floor reduction can be affected through girth-only conditioning if the degree distribution restricts the *highest* permissible variable node degree to be a relatively small number (*e.g.*, $d_v \leq 8$). The absence of high degree nodes, however, leads to threshold degradation. This is particularly true with codes of rate below 0.4 where density evolution has shown a significant fraction of high degree (*e.g.*, $d_v \leq 15$) nodes imperative if threshold is to approach Shannon capacity under iterative decoding.

In contrast to this, Ryan [54] advocates elimination of *low* degree nodes from the distribution. The irregular LDPC code in [2] also picked a distribution free of degree-2 variables. Though not explicitly described in [54] and [2], removal of low degree nodes from a distribution will raise overall graph EMD since low degree nodes can not substantially increase the EMD of any given cycle. Either rate or threshold is adversely affected by removing low-degree nodes since density evolution results indicate that a small fraction of low degree nodes can significantly raise code rate, while only marginally degrading threshold.

Density evolution assumes an infinite block length. Identifying optimal degree distributions for short block lengths remains an open area for research. This work assumes that asymptotically optimal degree distributions provide a good starting point for designing codes with short block lengths. Removal of low degree nodes from the distribution followed by girth-only conditioned graph construction by [54] is in fact implicity aligned with the proposed EMD graph construction. However, the ACE conditioning approach achieves similar error floors without resorting to removal of low degree nodes, which are known to play a role in rate maximization for a given threshold in the asymptotic case.



(a)



(b)

Figure 3.9. Results for (10000, 5000) codes. The BPSK capacity bound at $R = 0.5$ is 0.188dB.

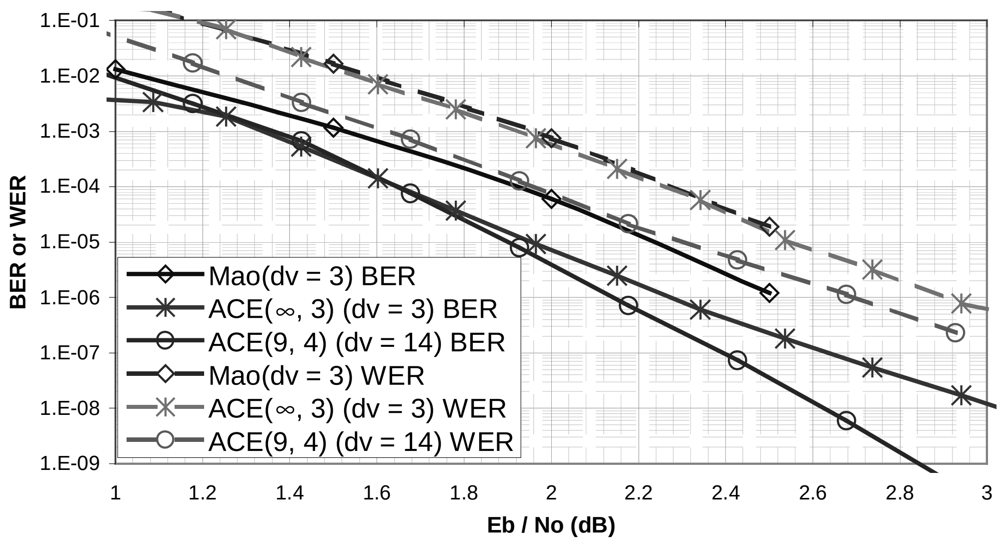


Figure 3.10. **Results for (1264, 456) codes.**

The BPSK capacity bounds at $R = 0.36$ is -0.394dB.

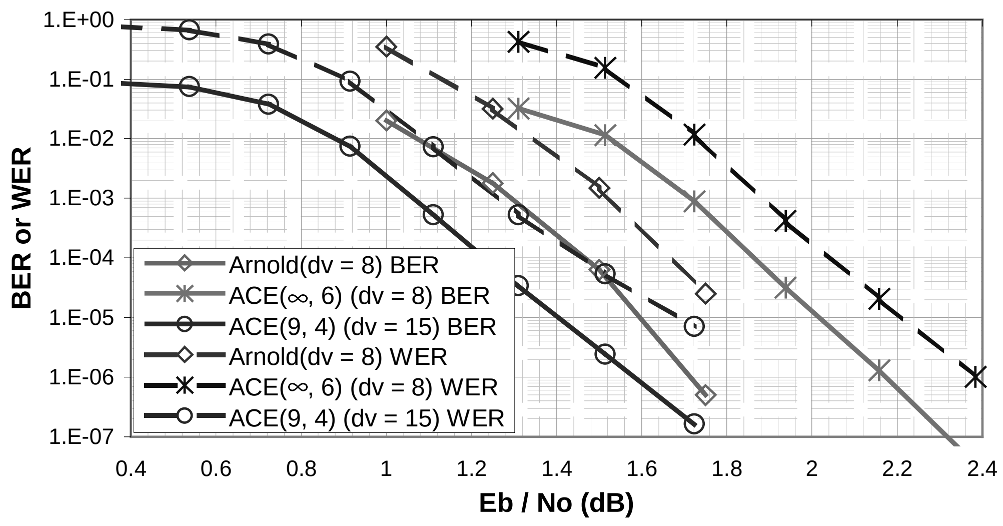


Figure 3.11. **Results for (4000, 2000) codes.**

The BPSK capacity bounds at $R = 0.5$ is 0.188dB.

Chapter 4

The Universality of Low-Density Parity-Check Codes in Scalar Fading Channels

Channel coding techniques that approach capacity for a large set of channel realizations, without specializing the transmission to the channel, are clearly desirable from complexity and system usability points of view. In the discussion that follows, a single code that can communicate reliably near the capacity of many different channels will be called "universal". A proof of the existence of codes that exhibit this property was provided by Root and Variaya in [39]. Root and Variaya's proof considered the compound channel that occurs when the actual channel is unknown to both transmitter and receiver but belongs to a set of possible channels known to both. Specifically, they proved that a single code exists that can communicate reliably over all channels $\mathbf{a}$ in the set $\mathbf{A}$ at rates arbitrarily close to the compound channel capacity given by,

$$C(\mathbf{A}) = \inf_{\mathbf{a}\in\mathbf{A}} (I(\mathbf{a})). \tag{4.1}$$

Where $I(\mathbf{a})$ is the mutual information induced by the transmitted power spectrum on the channel $\mathbf{a}$. For a given desired rate R, one might choose the set of channels $\mathbf{A} = \{\mathbf{a}|I(\mathbf{a}) > R\}$, such that the mutual information of every channel in the set is above the transmitted rate. In this way, Root and Varaiya's theorem says that "universal" or "robust" codes exist that support rate R over every channel where *any* code (with the specified transmit power spectrum) exists that supports this rate.

This chapter and the next provide an examination of carefully constructed LDPC codes in order to determine the degree to which they realize the promise of universal operation. The performance of these codes under two distinct types of channels, periodic fading channels and partial-band jamming channels, will be characterized. Each channel type is useful in its own way for describing the robustness properties of Low-Density Parity-Check (LDPC) codes.

To begin, period-p fading channels have at time i input x_i and output $y_i = a_{(i \bmod p)}x_i + n_i$, where n_i is additive white Gaussian noise (AWGN) with variance $N_o/2$ per dimension. The p-element vector $\mathbf{a} = [a_0\ a_1\ \ldots\ a_{p-1}]$ consists of complex scalars, which could be the subchannel gains of an Orthogonal Frequency Division Modulation (OFDM) system with p subcarriers. Nearly complete characterization of the performance of LDPC codes on channels with small p can be carried out experimentally and analytically through exhaustive parameterization of the fading vector $\mathbf{a}$. Analytic characterization will be provided via the periodic channel extension of Chung's Gaussian approximation [11] to density evolution. Robust operation in channels with longer periods is more difficult to completely characterize, though we do provide several specific examples. To further demonstrate the robustness with greater generality, we turn to the partial-band jamming channel.

The partial-band jamming channel occurs when a fraction of transmitted code symbols have a relatively poor signal-to-noise ratio (SNR) at the receiver (and are hence jammed) and another fraction experience a relatively good received SNR. It is usually the case that the selection of received signals that incur jamming versus those that do not is random (in adherence to the pre-determined proportions). Simulations performed such that jamming locations are varied from one codeword to the next provide a means for testing a large set of long period fading channels and of measuring the average fading performance of a code on this set. Good average performance over the set of channels is a necessary, but not sufficient, condition for the code to perform well on every channel individually as the Root and Varaiya result would predict. However, complete characterization of every partial-band jamming channel was beyond our computational ability.

Work on the design and characterization of universal channel codes has been conducted by Wesel *et al.* across several code paradigms for numerous channels. Initially, trellis codes for periodic fading channels were developed in [51][52][53]. Designs (for periodic channels) of universal serially concatenated turbo codes and LDPC codes (the first appearance of the present work) were performed in [23] and [6]. Universal constructions of space-time time trellis codes and diagonally layered space-time systems were presented in [22] and [33]. Finally, the robustness of LDPC codes in the generalized Gaussian (multi-input multi-output) channel has been presented in [5].

The next section provides mutual information definitions for the periodic fading channel. Section 4.2 discusses the design and operation of LDPC codes for the period-2 channel in detail. To demonstrate that robust performance is not limited to the period-2 channel, section 4.3 provides performance results for an LDPC code on four period-256 channels. A test of average performance on long periodic channels is made in section 4.4 using the partial-band jamming channel. Finally, conclusions from this work are drawn in section 4.5.

4.1 Mutual Information for Periodic Scalar Channels

The mutual information of the channel $y = ax + n$, where scale factor a is known at the receiver, can be expressed as,

$$I(X;Y,A) = I(X;A) + I(X;Y|A) = I(X;Y|A), \tag{4.2}$$

where X and A are independent (scalar) RVs. $I(X;Y|A) = E_A[I(X;Y|A = a)]$ is the expectation that defines the average mutual information of this channel if A is varying at the receiver. If, however, A is a deterministic constant ($A = a$) then the mutual information can be computed directly,

$$I(X;aX+N) = I(X;X+\frac{N}{a}) = h(X+\frac{N}{a}) - h(\frac{N}{a}). \tag{4.3}$$

The extension of this result to periodic fading follows for a particular instance of the p-element vector $\mathbf{a}$,

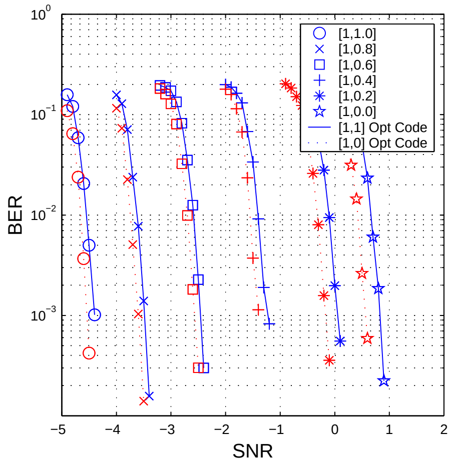
$$I(\mathbf{a}) = \frac{1}{p}\sum_{i=0}^{p-1} I(X_i; X_i + \frac{N}{a_i}), \tag{4.4}$$

which can also be used to define the capacity of a frequency selective fading channel in the context of OFDM modulation where i indexes the subcarriers. If both X_i and N are Gaussian and each X_i has the same average power, $E[\mathrm{X_i}^2] = E_x$, then the average mutual information over period p is,

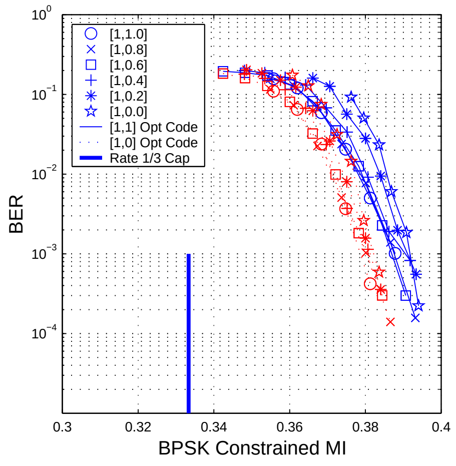
$$I(\mathbf{a}) = \frac{1}{p}\sum_{i=0}^{p-1} \log_2\left(1 + \frac{|a_i|^2 E_x}{2\sigma^2}\right). \tag{4.5}$$

The constant power constraint causes the mutual information in (4.5) to be less than the water-filling capacity that can be achieved if the transmitter knows $\mathbf{a}$. Nevertheless, Shannon's basic noisy coding theorem ensures that for each $\mathbf{a}$ there is a code with fixed symbol power E_x and rate R that achieves reliable communication with R arbitrarily close to $I(\mathbf{a})$. For example, p parallel Gaussian alphabet codes could be designed with the ith code assigned rate $R_i = \log_2\left(1 + \frac{|a_i|^2 E_x}{2\sigma^2}\right)$. This solution is unattractive as it requires transmitter and receiver to coordinate code selection depending on $\mathbf{a}$ and, of course, has tremendous complexity for large p .

In this work we turn to the broader result of Root and Varaiya [39] who proved that a *single* code exists that can communicate reliably at rates arbitrarily close to the compound channel capacity given by (4.1). While Shannon stated that for each channel there exists a code that provides reliable communication for that channel, Root and



(a)



(b)

Figure 4.1. (a) Code performance on the a $= [1, a]$ fading channel in terms of SNR. (b) Code performance on the a $= [1, a]$ fading channel in terms of Mutual Information. Dashed lines indicate operation of a code optimized for the a $= [1, 0]$ channel, solid lines indicate operation of a code optimized for the a $= [1, 1]$ channel.

Varaiya showed that for a given *set* of channels (collectively this set forms the compound channel) there is *a* code that provides reliable communication on all channels within this set. In the succeeding sections, simulation and density evolution results will show that a single LDPC code can perform with less than 0.1 bits of *excess* mutual information (per real signaling dimension) for compound channels where the cardinality of the channel set is large. Excess mutual information is defined as the capacity margin between the channel MI where the desired error probability is achieved and the information transmission rate R. We use excess mutual information as a performance measure throughout the remainder of the chapter as it provides as mechanism by which to directly compare channels that require dramatically different SNRs in order to support communication with a given code.

4.2 LDPC Performance on Period-2 Fading Channels

Two rate 1/3 LDPC codes are used for the simulations in Fig. 4.1. Each has (n, k)=(15000, 5000) and were realized from the degree distributions of Table 4.1. These degree distributions were found by constraining periodic density evolution for the $\mathbf{a} = [1, 1]$ (e.g. Gaussian) and $\mathbf{a} = [1, 0]$ (e.g. Gaussian with 50% erasure) channels and using an linear program (LP) solver to find the respective minimum threshold rate 1/3 codes with maximum left degree 15. As a benchmark, rate-1/3 BPSK constrained capacity is -5.3 dB (E_x/N_o). The codes were conditioned using the Approximate Cycle EMD (ACE) technique developed in [44], where EMD stands for extrinsic message degree. This graph construction technique is particularly attractive for use in conjunction with density evolution as it places no constraints on the underlying degree distribution of the code. The ACE technique is based on the idea of maximizing the multiplicity of "extrinsic" edges (meaning edges that do not participate in the cycle) that are connected to all cycles in the graph shorter than a given length. Such a construction improves the stopping set distribution in the graph (increases the mean stopping set size). Briefly, the reason for this is that stopping sets can be shown to be formed by closed clusters of cycles (e.g. cycles that are completely interconnected). Ensuring that each cycle (shorter than a given length) has at least a minimum number of external connections increases the average number of nodes required to form a closed cycle set. The codes used in this chapter have ACE parameters $d_{ACE} = 12$ and $\eta_{ACE} = 4$.

Figure 4.1 describes the performance of these codes in the period-2 channel with $\mathbf{a} = [a_0\ a_1]$, where $a_0 = 1$ and $a_1 = a$ (signaling is via BPSK modulation). Fig. 4.1(a) clearly shows that a decrease in a requires an increase in E_x/N_o (E_x/N_o describes the signal to noise ratio before channel scaling by a) to maintain constant BER. The plot versus SNR, however, does not provide an absolute view of the respective performance on each of the channels. Meaning, we might gather that performance on the $\mathbf{a} = [1, 1]$ channel is about 0.7dB away (at BER = 10^{-3}) from binary-input additive white Gaussian noise (BI-AWGN) capacity, however we can less easily determine if the code is performing as well on say the $\mathbf{a} = [1, 0.5]$ channel.

To form an absolute basis for comparison Fig. 4.1(b) plots the BER versus mutual information using (4.4) in the context of BPSK constrained signaling. The SNR of each plotted point in Fig. 4.1(a) has a corresponding mutual information (under the BPSK and periodic channel constraints), which yields the plot of BER versus mutual information given in Fig. 4.1(b). From this point of view, the robustness of the code to period-2 fading is apparent. When each of the five channels provides mutual information of at least 0.4 bits (i.e. 0.067 bits above the transmitted rate of 1/3) the codes communicate at or below BER=10^{-3}. Furthermore, the performance variation of both codes, on all channels, is less than 0.02 bits of mutual information. We say then, that these two codes are universal codes for period-2 fading since their mutual information requirement is essentially constant against channel variation. In the next section we discuss the difference between these two codes in terms of design and asymptotic threshold performance.

4.2.1 Code design for Period-p fading channels

To design a rate 1/3 code specifically for the $\mathbf{a} = [1, 0]$ channel, Chung's Gaussian approximation to density evolution is adapted to period-p fading channels and then used with $p = 2$. Following the assumption in [11] that the output of an individual variable or check node is Gaussian we proceed by expanding the argument to account for a known periodic fading vector $\mathbf{a}$.

At iteration l, degree-i variable nodes have their mean values, $m_{v,i}$, updated in correspondence to the periodic initial means given by $m_{a_j} = \frac{2a_j^2}{\sigma^2}$ and the means of messages arriving from constraint nodes (m_u),

$$m_{v,i}^{(l)}(j) = m_{a_j} + (i-1)m_u^{(l-1)}, j = \{0, \ldots, p-1\} \tag{4.6}$$

Randomly selected edges emanating from variable nodes adhere to the following Gaussian mixture density,

$$f_v^{(l)} = \sum_{j=0}^{p-1} \sum_{i=2}^{d_l} \frac{\lambda_i}{p} N\left(m_{v,i}^{(l)}(j), 2m_{v,i}^{(l)}(j)\right) \tag{4.7}$$

Where the outer summation mixes over the periodic fading vector and the inner summation mixes over the variable node edge-wise degree distribution. Following the constraint-node-update rule $\tanh\left(\frac{u}{2}\right) = \prod_{i=1}^{d_c-1} \tanh\left(\frac{v_j}{2}\right)$ we are interested in the expectation,

$$\begin{aligned} E[\tanh \frac{v^{(l)}}{2}] &= \int_{\Re} \tanh \frac{x}{2} f_x^{(l)} dx \\ &= \int_{\Re} \tanh \frac{x}{2} \left[\sum_{j=0}^{p-1} \sum_{i=2}^{d_l} \frac{\lambda_i}{p} N\left(m_{v,i}^{(l)}(j), 2m_{v,i}^{(l)}(j)\right)\right] dx \\ &= \frac{1}{p} \sum_{j=0}^{p-1} \lambda_1 \left(1 - \phi(m_{v,1}^{(l)}(j))\right) + \\ &\cdots + \frac{1}{p} \sum_{j=0}^{p-1} \lambda_{d_l} \left(1 - \phi(m_{v,d_l}^{(l)}(j))\right) \\ &= 1 - \frac{1}{p} \sum_{j=0}^{p-1} \sum_{i=2}^{d_l} \lambda_i \phi(m_{v,i}^{(l)}(j)) \end{aligned} \tag{4.8}$$

Where the function $\phi(x)$ (originally defined in [11]) equals $1 - E\left[\tanh \frac{u}{2}\right]$ if u is distributed as $N(x, 2x)$ (variance is fixed to $2x$ due to the symmetry condition).

Again from the constraint-node-updated rule, the expectation of a degree-k constraint node obeys,

$$E\left[\tanh \frac{u_k^{(l)}}{2}\right] = E\left[\tanh \frac{v^{(l)}}{2}\right]^{k-1} \tag{4.9}$$

which can be re-written as,

$$1 - \phi(m_{u,k}^{(l)}) = \left[1 - \frac{1}{p}\sum_{j=0}^{p-1}\sum_{i=2}^{d_l}\lambda_i\phi(m_{v,i}^{(l)}(j))\right]^{k-1} \tag{4.10}$$

and finally yields,

$$m_{u,k}^{(l)} = \phi^{-1}\left(1 - \left[1 - \frac{1}{p}\sum_{j=0}^{p-1}\sum_{i=2}^{d_l}\lambda_i\phi(m_{v,i}^{(l)}(j))\right]^{k-1}\right). \tag{4.11}$$

To complete the recursion, find the average mean value emanating from a constraint node via the average of the resulting Gaussian mixture which is given by,

$$\begin{aligned} m_u^{(l)} &= \sum_{k=2}^{d_r}\rho_k m_{u,k}^{(l)} \\ &= \sum_{k=2}^{d_r}\rho_k\phi^{-1}\left(1 - \left[1 - \frac{1}{p}\sum_{j=0}^{p-1}\sum_{i=2}^{d_l}\lambda_i\phi(m_{v,i}^{(l)}(j))\right]^{k-1}\right) \end{aligned} \tag{4.12}$$

substitution of terms in (4.6) for $m_{v,i}^{(l)}(j)$ gives a complete update recursion for $m_u^{(l)}$ from $m_u^{(l-1)}$.

The above recursion is linear in ρ instead of λ. An equivalent derivation that begins on the constraint rather than variable node side of the graph produces an update equation that is linear in λ and can be employed as a constraint in a linear program that seeks to find a rate maximizing λ given a set of initial means $m_{\mathbf{a}}$. In this form, $m_u^{(l)}$ is replaced by $r^{(l)}$ and it can be shown that $m_u^{(l)} \to \infty$ iff $r^{(l)} \to 0$.

$$\begin{aligned} r^{(l)} &= \tfrac{1}{p}\sum_{i=2}^{d_l}\lambda_i\sum_{j=0}^{p-1}\phi\left(m_{a_j} + (i-1)\sum_{k=2}^{d_r}\rho_k\phi^{-1}\left(1 - \left(1 - r^{(l-1)}\right)^{k-1}\right)\right) \\ r^{(l)} &= \sum_{i=2}^{d_l}\lambda_i\sum_{j=0}^{p-1}\tfrac{1}{p}h_i\left(m_{a_j}, r^{(l-1)}\right) \end{aligned} \tag{4.13}$$

The notation h_i is used as shorthand to represent the basis formed via the parameterization of $\phi(\cdot)$ and $r^{(0)} = \frac{1}{p}\sum_{\mathbf{a}} \phi\left(m_{a_j}\right)$, we state the LP formally as follows,

$$
\begin{aligned}
&\text{Min}\left(-\sum \lambda_i/_i\right)\\
&\text{s.t.}\\
&1^t\lambda = 1\\
&-I\lambda \preceq 0\\
&r > \sum_{i=2}^{d_l} \lambda_i \sum_{j=0}^{p-1} \frac{1}{p} h_i\left(m_{a_j}, r\right)\\
&0 < r < \frac{1}{p}\sum_{\mathbf{a}} \phi\left(m_{a_j}\right)
\end{aligned}
\tag{4.14}
$$

To obtain a code of a given rate (say 1/3) for the period-2 channel, an outer loop can be added to the optimization where bi-section on the initial means (the odd mean is fixed to zero due to the channel fade) is performed until the rate maximizing (λ, ρ) pair has rate 1/3. The columns in Table 4.1 labeled '$[1, 0]$' are the result of the optimization procedure when it is applied to the $\mathbf{a} = [1, 0]$ channel. Optimization results for the $\mathbf{a} = [1, 1]$ channel are also provided in the table.

Figure 4.2 provides asymptotic threshold results for the $\mathbf{a} = [1, 1]$ and $\mathbf{a} = [1, 0]$ optimized codes across $\mathbf{a} = [1, a]$ period-2 fading. As expected, the thresholds of each of these codes is best on the channel for which it was designed and worst on the opposite channel. The simulated performance of these two codes (dashed curves) across this parameterization is also provided in the figure (for BER = 10^{-3}). The simulated gap in the performance of the two codes on the $\mathbf{a} = [1, 0]$ channel follows closely from the gap predicted by density evolution (i.e. the $\mathbf{a} = [1, 1]$ code requires approximately 0.01 bits more MI, or about 0.25 dB more SNR, to achieve the same BER on the $\mathbf{a} =$

i	λ_i [1,1]	λ_i [1,0]	ρ_i [1,1]	ρ_i [1,0]
2	0.328189	0.354954	-	-
4	0.245943	0.249982	-	-
5	0.082931	0.065503	0.3000	0.5000
6	-	-	0.7000	0.5000
15	0.342935	0.329558	-	-

Table 4.1. Degree distributions optimized using Guassian approximation to density evolution adapted to periodic fading. Columns labeled $\mathrm{a} = [1, 0]$ indicate the distribution resulting from optimization for the period-2 channel where half of all received symbols are erased. Columns labeled $\mathrm{a} = [1, 1]$ indicate a period-2 code optimized for AWGN.

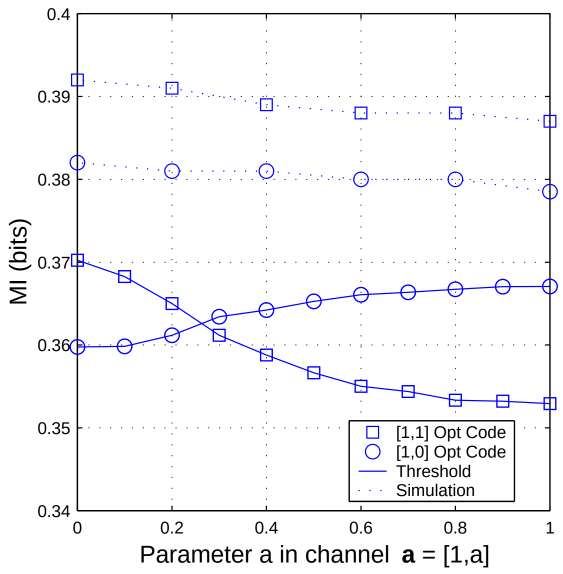


Figure 4.2. Mutual information thresholds of a $= [1, 1]$ and a $= [1, 0]$ optimized codes across a $= [1, a]$ fading (solid lines). Simulation results at BER = 10^{-3} for length 15,000 codes realized from the corresponding degree distributions.

$[1, 0]$ channel). However, as the channel becomes increasingly "white" the simulated performance of the $\mathbf{a} = [1, 0]$ optimized code does not become worse than that of the $\mathbf{a} = [1, 1]$ optimized code.

We note this result as interesting since the degree distribution for the $\mathbf{a} = [1, 0]$ code actually violates Chung's "stability" criterion for degree-2 nodes at SNRs for which it is operating at low BER on the $\mathbf{a} = [1, 1]$ channel. Said another way, threshold examination of the $\mathbf{a} = [1, 0]$ code on the $\mathbf{a} = [1, 1]$ channel yields constraint violations ($r < \sum_{i=2}^{d_l} \lambda_i \sum_{j=0}^{p-1} \frac{1}{p} h_i \left(m_{a_j}, r\right)$) near $r = 0$ (the stability region) at SNRs for which simulated operation yields reliable communication. The region near $r = 0$ corresponds to a point within the iterative decoding process where messages have achieved a high mean value and hence a low probability of error. For this reason, we hypothesize that code performance is less coupled to asymptotic predictions near $r = 0$ than in other regions ($r > 0$). Note also that the primary difference between the $\mathbf{a} = [1, 0]$ and $\mathbf{a} = [1, 1]$ optimized code degree distributions is the higher proportion of degree-2 nodes in the $\mathbf{a} = [1, 0]$ code. The fact that this code's actual performance on the AWGN ($\mathbf{a} = [1, 1]$) is better than that of the $\mathbf{a} = [1, 1]$ optimized code indicates that a higher proportion of degree-2 nodes than is indicated by density evolution should be assigned. Again, the stability criteria limits the number of degree-2 nodes due to violating constraints near $r = 0$ during optimization of the $\mathbf{a} = [1, 1]$ code. These empirical data seem to indicate that better codes can be found (at least for block lengths of 15,000 and below) through relaxation of the stability constraint. Such a relaxation is easily implemented, for instance, by raising the low end of the range for parameter r from 0 to α. Where α is a positive constant.

4.2.2 Robust Codes Vs. Optimally Matched Codes

If it is possible to design a code for the $\mathbf{a} = [1, 1]$ and $\mathbf{a} = [1, 0]$ channels, then the same is certainly true of all intermediate channels in $\mathbf{a} = [1, a]$. Such an exercise is useful to help measure the "cost" of robustness. Said another way, how much better would the performance of the system that customized the code to the channel be than the system that uses a single code for all channels.

The initial means for the even and odd phases of the $\mathbf{a} = [1, a]$ channel such that all

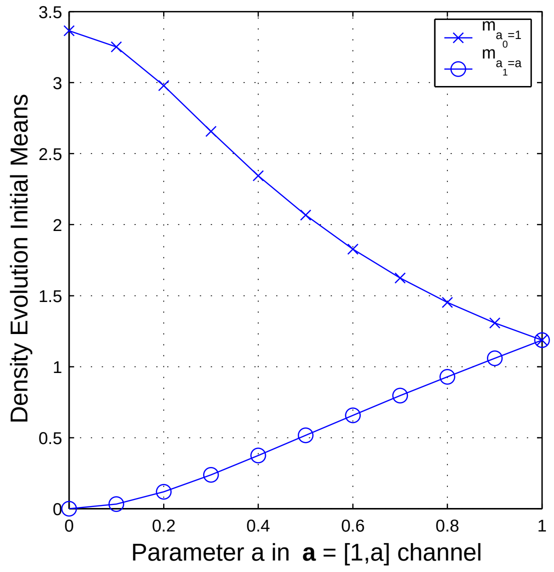


Figure 4.3. Density evolution initial means (for even/odd positions in a period-2 channel) that provide an aggregate mutual information of 1/3 bit.

channels across the parameter a have mutual information equal to 1/3 bits are shown Fig. 4.3. The means in Fig. 4.3 seed the periodic density evolution algorithm and a series of LPs are solved to find the rate maximizing degree sequence for each parameterization of the channel. Specifically, for a sequence of concentrated right degrees, whose average degree is decreasing, (concentrated right degree sequences were proven to be optimal in [11]) the rate maximizing left degree sequence was found via solution of an LP.

The rate maximization problem is convex in the parameterization of (concentrated) right degree. Therefore the concentrated ρ distribution that yields the overall rate maximizing λ in fact yields a globally optimal (λ, ρ) pair. The achievable rates, according to density evolution, on each of the ten fading channels (for left and right maximum degree constraints of 15) are given by the lower curve in Fig. 4.4. Note that the maximum possible rate achievable is 1/3 (due to the input MI constraint). The rate variation between the highest achievable rate (on the $\mathbf{a} = [1, 1]$ channel) and the lowest rate (on the $\mathbf{a} = [1, 0]$ channel) equals 0.007 bits. Such a result makes clear the fact that LDPCCs can be designed (if the period-2 channel is known in advance) so that essentially equivalent operation on each of the channels is possible. However, we compare the "flatness" of this result with the equally impressive asymptotic result of Fig. 4.2 where the $\mathbf{a} = [1, 0]$ optimized code has a threshold variation across $\mathbf{a} = [1, a]$ fading of approximately 0.008 bits. These density evolution results indicate that within the LDPC paradigm there is essentially no cost associated with the use of a single "universal" code versus a series of channel-specific codes in the context of the period-2 fading channel.

4.2.3 LDPC Period-2 Performance Compared to that of Serially Concatenated Convolutional Codes

LDPC represents one of several well known realizations of random linear codes with manageable decoding complexity. Parallel and serially concatenated convolutional codes also exhibit capacity approaching performance under AWGN channel conditions. Work similar to that in the present discussion, but that instead considers serial turbo codes, has been conducted in [23]. In the sections that follow we compare LDPC and serial turbo code performance under the period-2 and period-256 channelization scenarios.

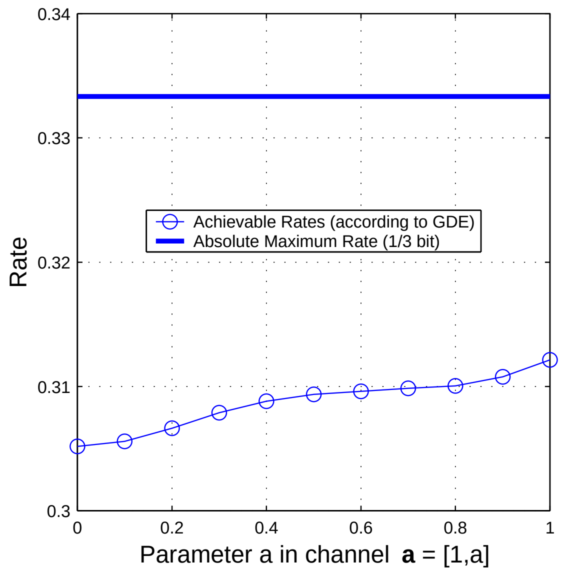


Figure 4.4. The maximum achievable rate for codes optimized for each instance in a parameterization of the a $= [1, a]$ channel. Mutual information due to initial means is held at 1/3 bit across the parameterization.

Figure 4.5 simultaneously plots excess MI and excess SNR for six different channel parameterizations under an 8PSK modulation constraint. Each of the lines emanating from the origin (one for each of the six channels) represents an absolute MI level of 1.0 bits on the abscissa (at origin) and on the ordinate (at origin) the absolute SNR level required to produce 1.0 bits of MI on the given channel. Points on the plot away from the origin measure MIs and SNRs in *excess* of capacity achieving levels. The differing slopes for each of these channels reveals why we have insisted on plotting performance versus MI throughout this chapter. Specifically, excess SNR (or SNR gap to capacity) varies with channel selection.

Also plotted in Fig. 4.5 is the performance of $\mathbf{a} = [1, 1]$ and $\mathbf{a} = [1, 0]$ optimized length 30,000, rate 1/3, LDPC codes on these six channels using Gray-labeled 8PSK modulation (therefore 10,000 total channel symbols). The performance of a rate 1/3 length 10,000 serial turbo code optimized for period-2 fading in [23] is also provided. It is important not to neglect scale in these plots. For instance the difference in the mutual performance of the $\mathbf{a} = [1, 0]$ optimized LDPC code across the channels is less than 0.05 bits. The serial turbo code exhibits a consistent excess SNR requirement that is striking, but has a variation in excess MI of 0.1 bits. The wider variation in excess MI of the serial turbo code comes from its exceptionally good performance on channels with low a values. The serial turbo code is strictly better than the $\mathbf{a} = [1, 1]$ optimized LDPC code and is very competitive with the $\mathbf{a} = [1, 0]$ optimized LDPC code.

As demonstrated in Figs. 4.1 and 4.2, length 15,000 LDPC codes under binary modulation (utilizing the same degree distributions as the length 30k codes of Fig. 4.5) exhibit an excess MI variation per real dimension (across the six period-2 channels) of less than 0.02 bits. A similar variation in excess MI per real dimension is observed with 8PSK modulation and block length 30,000.

4.3 Period-p Channels

To demonstrate that the code's robust operation in periodic fading is not limited to channels with small period p, consider the four period-256 channels in Fig. 4.6. These fading profiles were generated by realizing channels with 4, 8, and 16 multi-path components in the time domain. The time channels were randomly generated with each tap

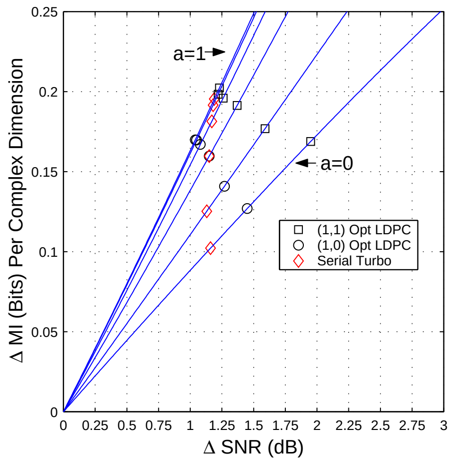


Figure 4.5. Mutual information and SNR in excess of that required for 1.0 bit per channel use on 8PSK in a $= [1, a]$ period-2 fading. Plotted points are operating points of two LDPC codes and a serial turbo code each of which is modulating 10,000 8PSK symbols per block at BER = 10^{-5}. Curves left to right indicate excess MI and excess SNR for a = $\{$1.0,0.8,0.6,0.4,0.2,0.0$\}$

magnitude and phase being a realization of a Rayleigh random variable. Exponential interarrival times between taps were assumed and an exponentially decaying envelope was imposed on the randomly realized taps. The 256-point fast-fourier-transform (FFT) of each of these channels was taken and the magnitude of the resulting FFT coefficients (OFDM subcarrier gains) are shown for each channel in the plot. Channel (d), is identical to channel (c), with the exception of the erasure of an arbitrarily selected block of 125 consecutive subcarriers.

The performance of the rate-1/3 blocklength 15,000 $\mathbf{a} = [1, 1]$ optimized LDPC code on these channels using QPSK modulation, where even(odd) code bits are mapped

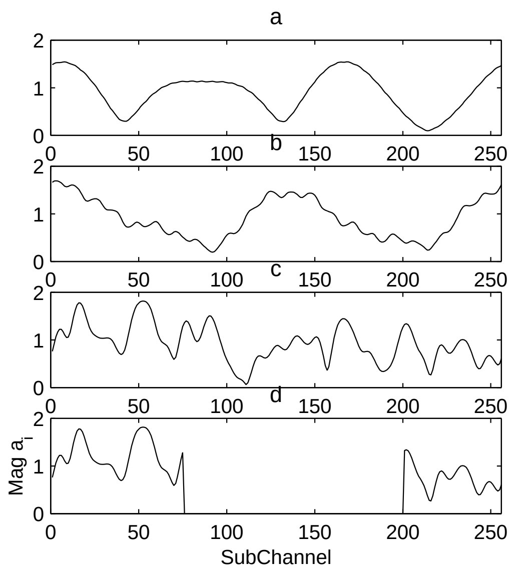


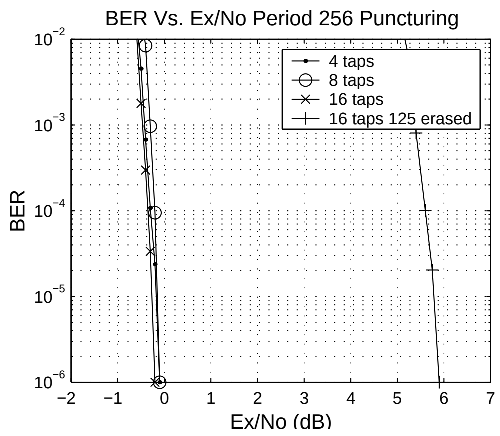
Figure 4.6. Four period-256 Fading Channels.

to I(Q) components, is given in Fig. 4.7(a). As a manifestation of the law of large numbers on the random subcarrier mutual informations [10], the QPSK constrained mutual information for these three channels is very similar. This explains the proximity of the three BER vs. SNR curves. Because of severe erasure distortion, channel (d) requires much more SNR for a given BER than the other channels to achieve the same level of mutual information. However, Fig. 4.7(b) shows that from the mutual information point of view the code works virtually as well on channel (d) as on channels (a,b,c).

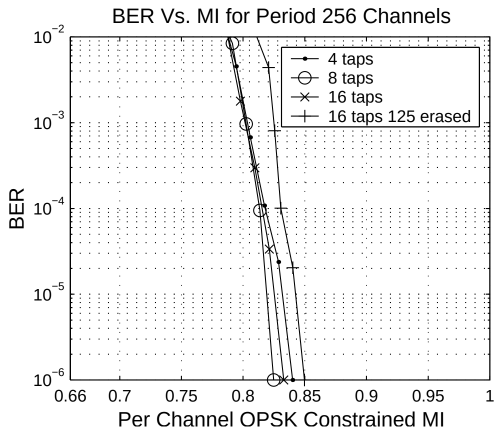
At first glance, it may seem surprising that the code can communicate with 125 of the 256 subcarriers completely erased. However, the supremum of erasure rates for this code on the BEC channel,

$$\varepsilon^* = \sup(\varepsilon = x_0 | x_l = x_0\lambda(1 - \rho(1 - x_{l-1})) \to 0, l \to \infty)$$

has $\varepsilon^* = 0.613$. Note that ε^* is an asymptotic measure that can only be achieved in the



(a)



(b)

Figure 4.7. (a) Code performance on four period-256 fading channels. (b) Code performance on four period-256 fading channels in terms of MI.

limit of infinite block length. For the length 15,000 code used in this simulation (the $\mathbf{a} = [1, 1]$ optimized code), $\varepsilon^* = 0.59$ was found via simulation. Thus the minimum capacity of the QPSK BEC channel on which this code can be expected to communicate reliably is given by $C_{BEC} = 2(1 - \varepsilon^*) = 0.82$. The high SNR (erasure) capacity of channel (d) is equal to $2(1 - (125/256)) = 1.02$ bits. Therefore, it is reasonable to expect that the code can operate on this channel when E_x/N_o is large. However, we emphasize the more remarkable result that the difference in mutual information required for the code to operate on each of these four very different period-256 channels is less than 0.025 bits.

As described in [50], the serial turbo code can perform well on channels (a),(b), and (c) in Fig. 4.6 through the use of a random channel interleaver, but fails to provide reliable communication at any SNR on channel (d) (the %50 erasure channel) unless the interleaver is "matched" to the channel. To date we know of no coding methodology, other than LDPC, that can communicate robustly (meaning without the augmentation of a matched channel interleaver) on channels such as (d).

4.4 LDPC Performance on the Partial-Band Jamming Channel

The partial-band jamming channel model used in the results that follow is that same as the one previously described in [35] and [20]. We limit our discussion to the case of coherently detected BPSK modulation under a frequency hopped scenario in which a fraction ρ of the available channels are jammed. All of the channels experience additive thermal noise due to the receiver front end. The SNR of this noise is fixed to $E_b/N_o = 20$ dB so as to be consistent with results in [35]. Channels that are jammed, however, also incur the addition of band-limited white Gaussian noise with power spectral density $\rho^{-1}N_I$ over a fraction ρ of the band. The total jamming noise power $\rho(\rho^{-1}N_I) + (1-\rho)0$ is equal to N_I, but is independent of ρ. Bit energy to interference ratio, E_b/N_I is the most common measure of performance on this channel. Perfect channel state information has been assumed for the LDPC results that will be presented. This implies that very low values of E_b/N_I tend to make jammed channels look like erasures as the log-likelihood ratios computed from channel observations are inversely scaled by the noise variance in a given subchannel. On the other hand, as ρ is increased to unity (where all subchannels are jammed), the channel begins to appear much like a standard AWGN channel.

Fig. 4.8 provides simulation results for two rate 1/3 LDPC codes. Both are realized from the degree sequence of the $\mathbf{a} = [1, 0]$ optimized code described in Table 4.1. The first code has length 4096 and the second length 15,000. The performance of a length 4096 turbo product code with comparable rate [35] is also provided. An important parameter for code performance on the partial-band jamming channel is the so-called dwell interval. This quantity describes the number of successive code symbols that will be transmitted on a given sub-channel before the modulation is hopped to another sub-channel. For sake of comparison with results in [35] we have fixed the dwell interval to 32 for the length 4096 code and to 30 for the length 15000 code. We have also made the assumption that channels are “framed” around single code words. This implies that for the length 4096 code there are 128 subchannels and $\lceil \rho 128 \rceil$ of these will be jammed. There are 500 subchannels per frame for the length 15000 code. The distribution of jammed subchannels is realized uniformly and independently from one codeword transmission to the next. This technique is meant to yield an average

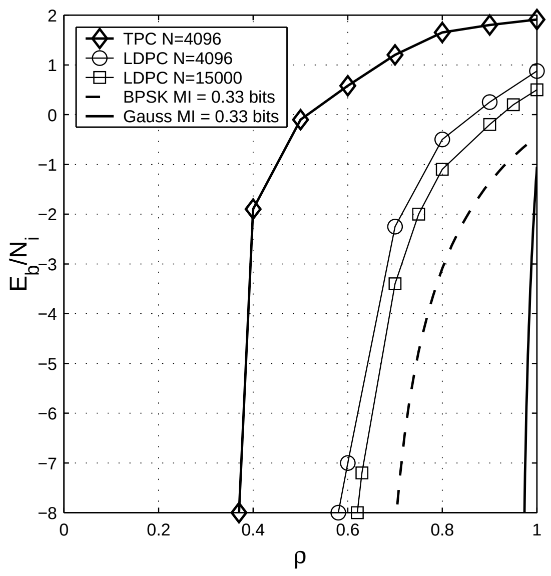


Figure 4.8. Performance of Rate 1/3 LDPC codes with blocklength 4096 and 15,000 on the partial-band jamming channel compared to a block-length 4096 turbo product code. E_b/N_i vs. ρ curves that maintain a constant Gaussian signaling capacity (MI) and BPSK constrained capacity (cMI) of 1/3 of a bit are also displayed. FER = 10^{-3} for the three simulated curves.

jamming result for a given code across a parameterization of ρ and E_b/N_o.

Constant mutual information (MI) curves for the partial-band jamming channel are also included in Fig. 4.8. To compute these curves consider the MI level in partial-band jamming,

$$MI = \rho f\left(SNR_J\right) + (1-\rho) f\left(SNR_{NJ}\right), \tag{4.15}$$

where SNR_J defines the symbol signal to noise ratio in the jammed subchannels and SNR_{NJ} defines the symbol signal to noise ratios in the non-jammed subchannels. In the case of Gaussian signaling, $f(x) = \log_2(1 + x)$, and for the BPSK constrained case $f(x)$ is evaluated via numerical integration. In the partial-band jamming simulations performed for this chapter, SNR_{NJ} is held fixed at a level which corresponds to $E_b/N_o = 20$dB. In the unconstrained case the term $\log_2(1 + SNR_{NJ})$ is therefore a constant (η) which can be determined via solution to the equation $\eta = \log_2(1 + \eta E_b/N_o)$, which is $\eta = 9.96$ bits (for $E_b/N_o = 20$ dB). In the BPSK constrained case $MI_{BPSK}(SNR_{NJ})$ saturates to $\eta = 1$ bit at this high SNR.

We are interested in values of (ρ, SNR_J) that yield constant levels of mutual information. We therefore fix the MI to some constant level, say $1/3$ of a bit. If we also fix ρ, it is possible to uniquely determine SNR_J (analytically for unconstrained and via table lookup for the BPSK constrained case). The resulting SNR_J can then be converted to E_b/N_I via the following relations,

$$\begin{aligned} SNR_J &= \frac{E_s}{\frac{N_I}{\rho} + N_o} = \frac{1}{\frac{1}{\rho R \frac{E_b}{N_I}} + \frac{1}{R \frac{E_b}{N_o}}} \\ \frac{E_b}{N_I} &= \frac{SNR_J}{\rho R \left(1 - \frac{SNR_J}{R \frac{E_b}{N_o}}\right)}. \end{aligned} \tag{4.16}$$

A large discrepancy can be observed between the BPSK-constrained and Gaussian-signaling mutual information curves in Fig. 4.8. This is due primarily to the fact that the non-jammed subchannels provide far more mutual information (9.96 bits) than can be provided by BPSK modulation, which in turn implies that with Gaussian signaling, just a small fraction of the subchannels need to be non-jammed for the expected mutual information in the channel to reach $1/3$ of a bit. We note that a system that achieves an average spectral efficiency of $1/3$ of a bit, and that approaches the unconstrained Gaus-

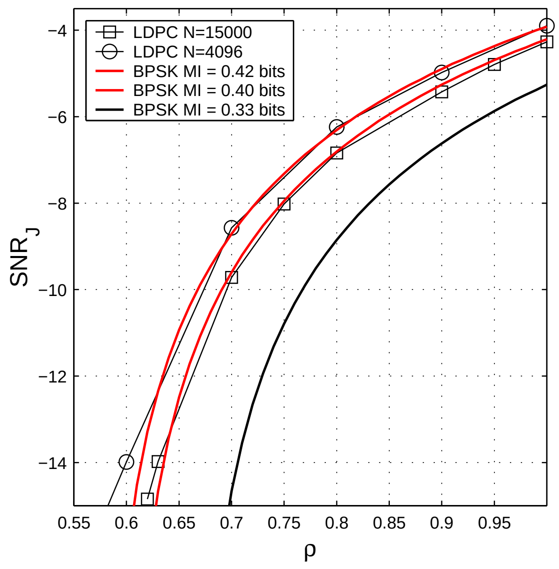


Figure 4.9. SNR,ρ performance of length 4096 and 15000 LDPC codes compared to SNR, ρ levels required to achieve 0.42, 0.4, and 1/3 of a bit of mutual information. FER = 10^{-3}

sian capacity, can be achieved by simultaneously increasing modulation cardinality and decreasing code rate. For instance a rate 1/6 code driving QPSK can be expected to perform better than the rate 1/3 BPSK system.

The curves in Fig. 4.8 represent systems with rate-$1/3$ or contours of constant mutual information of 1/3 of a bit. Fig. 4.9 plots different constant mutual information curves in terms of an absolute SNR_J ordinate. This avoids comparing E_b/N_I terms that differ only because of different rate (since $\frac{E_b}{N_I} \simeq SNR_J - 10\log_{10}(\rho R)$).

When the partial-band jamming channel provides 0.4 bits of mutual information, the length 15,000 LDPC code (the $\mathbf{a} = [0, 0]$ optimized code) operates with an FER = 10^{-3} at all but the lowest values of ρ. The same can be stated for the length 4096 code when the channel supports 0.42 bits of mutual information. Restating this result, the length 15000 code provides reliable communication when the excess MI in the channel is $\Delta MI = 0.4 - 1/3$, or roughly 0.067 bits. We note that the performance of the coded system on parameterizations of the channel that correspond to “erased” cases (ρ values near the erasure capacity of the code) are slightly inferior to the performance observed in an AWGN ($\rho = 1$) parameterization. The closeness with which the simulated performance tracks constant contours of mutual information in the figure provides clear empirical evidence of code robustness across an extremely broad range of channels. Finally, we note that the MI level (e.g. 0.4 bits) required for reliable communication in this channel is comparable to the levels required by the period-2 and period-256 fading channels.

4.5 Conclusion

In this work, we have taken a mutual information, rather than an SNR, approach to measuring code performance over periodic Gaussian and partial-band jamming channels. Root and Varaiya showed that a single code exists that can communicate reliably on all of the channels in a given set provided that the rate of the code is less than the smallest mutual information of all channels in the set. It has been shown for a quantized spread of all period-2 channels and for several arbitrarily selected period-256 channels that LDPC codes are an incarnation of Root and Varaiya’s promise of “universal” codes. We have also described and used periodic density evolution to design codes matched to channels and to determine the thresholds of existing codes across parameterizations of the $\mathbf{a} = [1, a]$ channel. Root and Varaiya's theorem applies to any particular instance of the partial-band jamming channel. However, we have averaged the performance of a given code across many thousands of instances of the PBJ channel in order to test the universality of the codes across a large sampling of channels. While it is true that the performance of the codes on some particular PBJ channel may have been poor (and this event went undetected due to the averaging process), we have nevertheless shown that the average excess mutual information requirements of the codes on this channel are very similar to those of the codes on the periodic fading channels.

Chapter 5

The Universal Operation of LDPC Codes in Vector Fading Channels

A compound channel occurs when the actual channel is unknown to both transmitter and receiver but belongs to a set of possible channels known to both. Root and Varaiya's compound channel theorem [39] applied to the linear Gaussian vector channel,

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n} \qquad \mathbf{n} \sim \mathcal{N}(\mathbf{0}, N_0\mathbf{I}_n), \tag{5.1}$$

indicates that for a given rate R and input distribution there exists a *single* code that can achieve reliable information transmission at rate R on every channel $\mathbf{H}$ for which the input distribution induces a mutual information (MI) higher than R. The immediate implication of this result is that good error performance on one particular channel does not have to come at the expense of significant performance degradation on others. Codes that have consistently good proximity to capacity (to the extent their blocklength and decoding complexity permit) over a class of vector-input channels will be referred to in this chapter (and the last) as *universal* codes. Since the linear Gaussian vector channels are commonly called space-time channels today, we will call to such codes universal space-time codes.

The capacity promise of multiple-input multiple-output (MIMO) systems in rich scattering environments [15] makes the existence and use of universal codes of practical interest. In [53] Wesel et. al constructed universal trellis codes for periodic erasure channels. The universal property of LDPC codes in the context of periodic fading channels was described by Jones et. al in [6]. In [21][22] Köse and Wesel found by exhaustive search universal space-time trellis codes for the 2×2 linear Gaussian vector channel.

In this chapter we demonstrate that LDPC codes are universal over a class of matrix channels by showing that bit-multiplexed LDPC coding on the MIMO channel yields essentially constant per dimension excess mutual information performance (around 0.1 bits per real dimension) for all the channels that we examine. For a matrix channel, the excess mutual information per dimension is defined as the capacity margin between the operational channel MI per transmit antenna and the information transmission rate per transmit antenna.

Iterative-detection-and-decoding methods for MIMO systems that are based on the turbo decoding principle are often referred to as Turbo-BLAST [40] [41][46]. Several techniques for low-complexity, high-performance iterative detection and decoding receivers are available in the literature. In [40] and [41] the authors introduce a sub-optimal receiver based on a minimum-mean square-error (MMSE) soft interference cancellation detector. In [43] a "list" sphere decoder is used to iteratively detect and decode either simple convolutional or more powerful turbo codes. In [1] the authors present an iterative-greedy demodulation-decoding technique for turbo codes based on a greedy detection method for multiuser communications. More recently in [27] the authors present a low-density parity-check (LDPC) coded MIMO OFDM system using either the optimal soft maximum *a posteriori* (MAP) demodulator or the low complexity minimum-mean square-error soft interference cancellation (MMSE-SIC) demodulator. The results reported in [27] show that a system based on the MMSE-SIC detector suffers a performance degradation in comparison to a system based on the MAP detector. In our work however, we have observed very little difference in the performance of these two detection schemes for systems with a small number of transmit and receive antennas.

We demonstrate LDPC code robustness on the $2{\times}2$ MIMO channel via exhaustive parameterization. We then introduce and discuss the complexity of various detection techniques. In particular, we consider the soft MAP detector, the MMSE-SIC detector, and we introduce a simple MMSE suppression detector and a minimum-mean square-error *hard* decision interference cancellation (MMSE-HIC) detector. We compare the performance and complexity of these schemes and find that the very low complexity MMSE suppression and MMSE-HIC detectors have only a small degradation in performance (less than 0.5 dB) on fast fading Rayleigh channels. In the final section of the chapter we will show that the excess mutual information requirement for a given code, on a per real-dimension basis, is independent of system configuration ($2{\times}2$, $3{\times}3$, $4{\times}4$, $8{\times}8$) in Rayleigh fast fading.

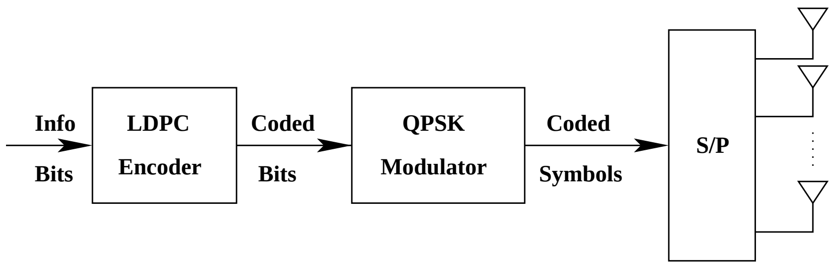


Figure 5.1. Transmitter structure of an LDPC coded BLAST system.

5.1 Excess Mutual Information as a Measure of Performance

The system model under consideration is an LDPC-coded MIMO system with n_T transmitter antennas and n_R receiver antennas, signaling through frequency-nonselective fading. The transmitter structure is illustrated in Fig. 5.1. The information data is first encoded by an LDPC code, modulated by a complex constellation with 2^{M_c} possible signal points and unit average energy, and then distributed among the n_T antennas. Let $\mathbf{x}$ be an $n_T \times 1$ vector of transmitted symbols with components $x_1, x_2, \ldots, x_{n_T}$ and $\mathbf{y}$ an $n_R \times 1$ vector of received signals with components $y_1, y_2, \ldots, y_{n_R}$, related by

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n} \tag{5.2}$$

where $\mathbf{H} = [\mathbf{h}_1 \mathbf{h}_2 \ldots \mathbf{h}_{n_T}]$ is the $n_R \times n_T$ complex channel matrix known perfectly to the receiver, and $\mathbf{n}$ is a vector of independent zero-mean, complex Gaussian noise entries with variance $\sigma^2 = N_0/2$ per real component. We assume that the average signal-to-noise ratio (SNR) at each receiver antenna, denoted by ρ, is independent of the number of transmitter antennas n_T.

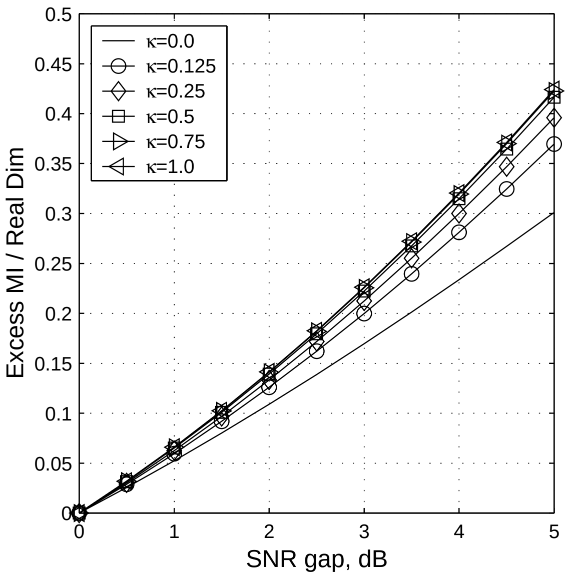


Figure 5.2. Excess MI per real dimension vs. SNR gap for 2×2 matrix channels, $R = 4/3$ bits/channel use, $\lambda_1 = 1$, for eigenvalue skews (top to bottom) $\kappa = 1, 0.75, 0.5, 0.25, 0.125, 0$.

We note that mutual information associated with the channel model in (5.2) identifies the fundamental information-carrying potential of the channel for a specific input distribution which in this work will be a uniform power distribution over a finite constellation that is modulating each antenna element. A universal code should provide performance that is consistent in terms of required excess mutual information. The common way to plot BER performance is versus channel signal-to-noise ratio (SNR). Since MI on the additive white Gaussian noise (AWGN) channel is a monotonic (and almost linear) function of SNR in dB, $MI_{\text{Gauss}} = \log_2(1 + \text{SNR})$, this representation is essentially equivalent to plotting BER against MI. For a fixed transmission rate R, SNR gap is defined as the difference between the SNR required to achieve the desired

BER and the SNR at which the channel capacity in (5.3) is equal to R bits per channel use. However, when assessing the performance of a code over a variety of linear Gaussian channels, considering SNR performance or SNR gap is problematic because the monotonic relationship between SNR in dB and MI is different for different channel eigenvalue skews.

To better understand the previous statement, consider a 2×2 linear Gaussian vector channel $\mathbf{H}$ and its MI under the assumption of uniform power distribution and a Gaussian signaling alphabet across antennas,

$$\begin{aligned} MI(\mathbf{H}) &= \log_2\left(1+\frac{SNR}{2}\lambda_1\right)\left(1+\frac{SNR}{2}\lambda_2\right) \\ &= \log_2\left(1+\frac{SNR}{2}\lambda_1\right)\left(1+\frac{SNR}{2}\kappa\lambda_1\right). \end{aligned} \tag{5.3}$$

Where λ_1 and λ_2 are the eigenvalues of $\mathbf{H}\mathbf{H}^\dagger$, $\kappa = \lambda_2/\lambda_1$ is the eigenvalue skew and ρ is the average SNR per receive antenna.

Fig. 5.2 illustrates the excess MI per real dimension as a function of the SNR gap in dB for $R = 4/3$ bits/channel use and different eigenvalue skews κ. Note that the excess MI curves are approximately linear functions of the SNR gap in dB, however the slope depends on the eigenvalue spread (eigenskew) of the channel. In other words, a constant level of excess MI is achieved by differing excess SNR levels (depending on the eigenskew of the channel). The MI available in the channel is the absolute measure for performance, while an excess SNR measure depends both on the MI level and the specific channel realization (eigenskew).

5.2 LDPC MI Performance Under MAP Detection on 2×2 Channels

In this section the performance of two different Rate 1/3 length-15,000 bit-multiplexed LDPC codes modulating QPSK (for a net rate of 4/3 bits per transmission) on parameterized 2×2 MIMO channels will be described. The Maximum A Posteriori (MAP) detector will be used throughout this section and the reader is referred to section 5.3 for a complete description of the operation of this and other detection techniques.

Each of the 2×2 channels will be characterized by three parameters: two rotation parameters and the ratio of the eigenvalues in the system (or eigenskew). The value of such a parameterized assessment is that 'worst' and 'best' case channels can be identified found since channel matrix averaging, which occurs in both fast and quasi-static Rayleigh fading experiments, is explicitly avoided. Furthermore, 'flatness' of the excess mutual information measure versus channel skew becomes a criteria for comparing the degree of robustness possessed by a given code. Again, in our terminology, a code is universal if reliable communication (for instance BER = 10^{-5}) occurs at the same (small) excess mutual information level across all channels. Of course, the absolute SNR required to achieve a given mutual information level will vary with the degree of fading imposed by the channel matrix. A description of the channel sampling problem is given as follows,

$$\mathbf{H} = \sqrt{\lambda_1} \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{\kappa} \end{bmatrix} \begin{bmatrix} \cos(\phi) & \sin(\phi)e^{j\theta} \\ -\sin(\phi)e^{-j\theta} & \cos(\phi) \end{bmatrix} \tag{5.4}$$

where $\kappa = \lambda_2/\lambda_1$ is the eigenvalue skew of the Hermitian matrix $\mathbf{HH}^\dagger$, $\phi \in [0, \pi/2]$, and $\theta \in [0, 2\pi]$. In this chapter we sample the above matrix via the parameters $\kappa = \{0, 0.125, 0.25, 0.5, 0.75, 1\}$, $\phi = \{0, \pi/4\}$ and $\theta = \{\pi/4, 3\pi/4\}$. Performance on each channel in this sampling is measured by the mutual information in excess of the transmission rate of $4/3$ bits/channel-use required to achieve a BER of 10^{-5}. Note that the product of a diagonal eigenmatrix and a complex Givens rotation does not parameterize *every* 2×2 matrix, but does parameterize every interesting 2×2 matrix. Meaning that all H matrices not described by (5.4) are in fact isomorphs (with respect to the receive array) of a matrix that is described by (5.4).

Before discussing the performance across these channels we draw an analogy to periodic fading SISO channels. Note that a period-2 SISO fading channel, $y_k = a_{(k \bmod 2)} x_k + n_k$, with fading vector $\mathbf{a} = [\sqrt{\lambda_1}, \sqrt{\lambda_2}]$ is equivalent to a diagonal ($\phi = 0$) 2×2 matrix fading channel (but requires two channel uses to relay the same information). Following the work in [19], we are able to optimize LDPC degree distributions for period-p fading via an adaptation of the Gaussian approximation to density evolution. In specific, at iteration l, degree i variable nodes have their mean values updated in correspondence to the periodic initial means given by $m_{a_j} = \frac{2a_j^2}{\sigma^2}$ (where a_j are known fading gains) and the means of messages arriving from check nodes (m_u),

$$m_{v,i}^{(l)}(j) = m_{a_j} + (i-1)m_u^{(l-1)}, j = \{0, \ldots, p-1\}. \tag{5.5}$$

Randomly selected edges emanating from variable nodes adhere to the following Gaussian mixture density,

i	λ_i [1,1]	λ_i [1,0]	ρ_i [1,1]	ρ_i [1,0]
2	0.27603	0.354954	-	-
3	0.11195	-	-	-
4	0.17229	0.249982	-	-
5	0.01712	0.065503	-	0.5000
6	-	-	1.0	0.5000
15	0.42261	0.329558	-	-

Table 5.1. Degree distributions optimized using Guassian approximation to density evolution adapted to periodic fading. Columns labeled $\mathbf{a} = [1, 0]$ indicate the distribution resulting from optimization for the period-2 channel where half of all received symbols are erased. Columns labeled $\mathbf{a} = [1, 1]$ indicate a period-2 code optimized for AWGN.

$$f_v^{(l)} = \sum_{j=0}^{p-1} \sum_{i=2}^{d_l} \frac{\lambda_i}{p} \mathcal{N}\left(m_{v,i}^{(l)}(j), 2m_{v,i}^{(l)}(j)\right) \tag{5.6}$$

Using this kernel, codes optimized for $\mathbf{a} = [1, 1]$ and $\mathbf{a} = [1, 0]$ period-2 fading channels were designed and tested across the parameterization of 2×2 channels. The resulting degree distributions are given in Table 5.1.

5.2.1 Gaussian, Constellation Constrained, and Parallel Independent Decoding Mutual Information

We next describe the mutual information measures that can be considered when converting an SNR/BER point in a given simulation to an MI/BER point. Of course, we have already stated that the mutual information between transmitter and receiver using a Gaussian codebook follows $\log_2\left(1+\frac{SNR}{2}\lambda_1\right)\left(1+\frac{SNR}{2}\lambda_2\right)$ and is independent of Givens rotation parameters.

In general, however, two other capacity measures are relevant to a system that modulates an LDPC code directly onto a channel and these measures do depend on the channel rotation. The first is the modulation constrained channel capacity $I(Y,H;X)$ and the second is the Parallel Independent Decoding (PID) capacity [48]. The operational capacity of the LDPC coded system is bounded between these two capacities,

$$\begin{gathered} I(Y,H;X) \geq \\ \sum_{i=0}^{N_t M_c - 1} I(Y,H;X^{b_i}|\tilde{X}^{b_0},...,\tilde{X}^{b_{i-1}},\tilde{X}^{b_{i+1}},...,\tilde{X}^{b_{N_t M_c - 1}}) \geq \\ \sum_{i=0}^{N_t M_c - 1} I(Y,H;X^{b_i}) \end{gathered} \tag{5.7}$$

Proof that operating capacity of the system is lower bounded by $\sum_{i=0}^{N_t M_c - 1} I(Y,H;X^{b_i})$ is given as an appendix. This is the mutual information that is available in a system without feedback between the decoder and detector. In the case of Gray labeling on a SISO channel, the inequalities in (5.7) are sharp (practically speaking) and the operational capacities of bit multiplexed LDPC systems are congruent with constrained channel capacity. If system design dictates that the operational capacity be equal to constrained channel capacity, then a hybrid approach can be used which modulates separate LDPC codewords onto Gray labeled constellations at each transmit antenna [26]. The receiver then performs multi-stage decoding where each decoder uses all channel observation vectors in conjunction with soft or hard information from the other decoders in order to determine the set of modulated codewords. The operational capacity of such a system does not provably equal channel capacity, but for the case of Gray labeled modulation it was shown in [26] to be negligibly smaller. For the $2{\times}2$ parameterized system, Guassian input alphabet, constrained (net capacity), and PID capacities are given in Fig. 5.3.

In the figure, for each eigenskew, the SNR level that yields 4/3 bits when $\phi = 0$ (for the Net and PID cases) is used across the span of considered ϕ values. This particular ϕ was chosen because $\phi = \{0, \pi/2, \pi\}$ all yield diagonal channels for which the Net and PID capacities are immeasurably different at any given SNR. Channels with $\phi = \pi/4, 3\pi/4$ maximize Net (constrained) capacity. In fact, with these channels we observe that the Net capacity approaches the Gaussian alphabet capacity closely (which is reasonable given the relatively low rate loading of the QPSK constellations). Finally note that as κ approaches unity, the Net and PID mutual informations do not vary with parameter ϕ.

The robustness results for the $\mathbf{a} = [1, 0]$ optimized code are given in Fig. 5.4. Following the legend, the first three curves measure excess mutual information in terms of difference between the rate of the code (4/3 bits) and the mutual information that is supplied between a transmitter and receiver (assuming a Gaussian codebook) at SNRs

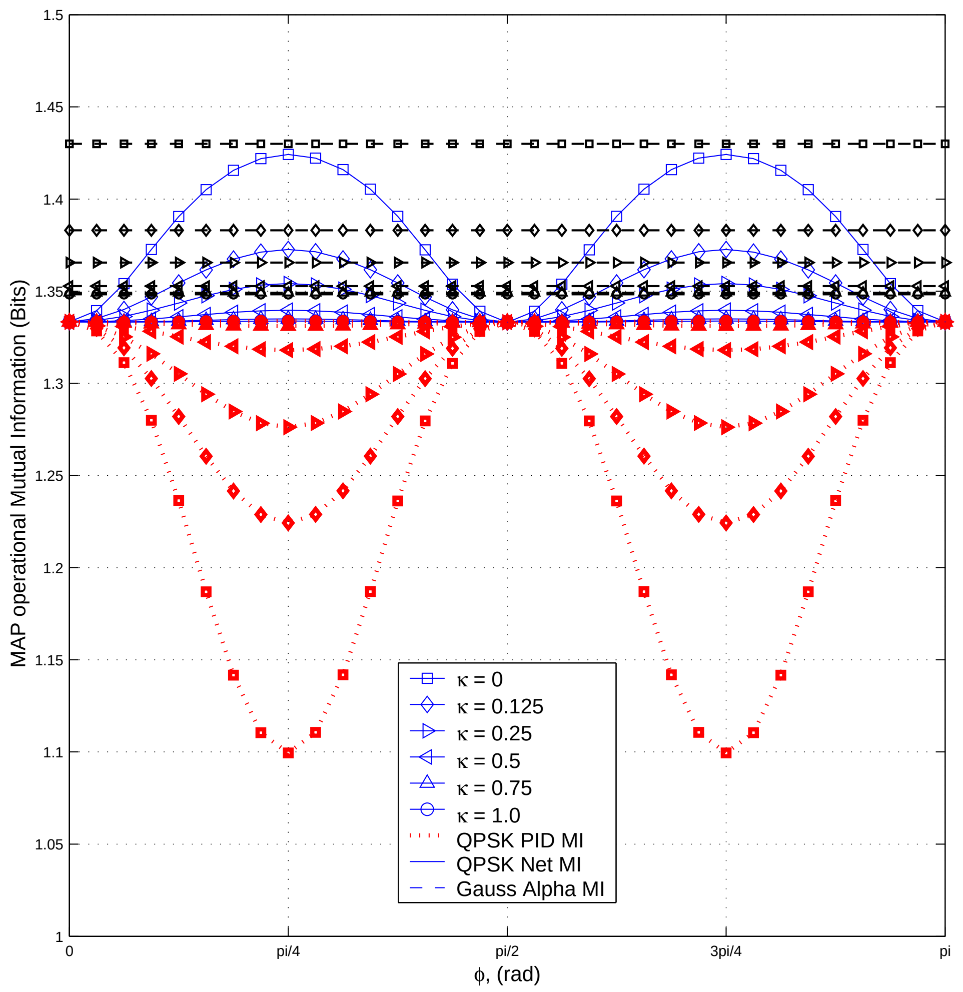


Figure 5.3. Channel mutual information versus channel matrix parameter ϕ and eigenskew. Gaussian Alphabet, QPSK modulation (net), and PID decoding capacities are shown. For each eigenskew, the SNR level that yields 4/3 bits when $\phi = 0$ (for the Net and PID cases) is used across the span of considered ϕ values. Note that at $\phi = \pi/4, 3\pi/4$ that the Net capacity is maximized (and nearly equals Gaussian Alpha capacity) and the PID capacity is minimized. When $\phi = 0, \pi/2, \pi$ (diagonal channels) Net and PID capacities are immeasurably different for a given SNR.

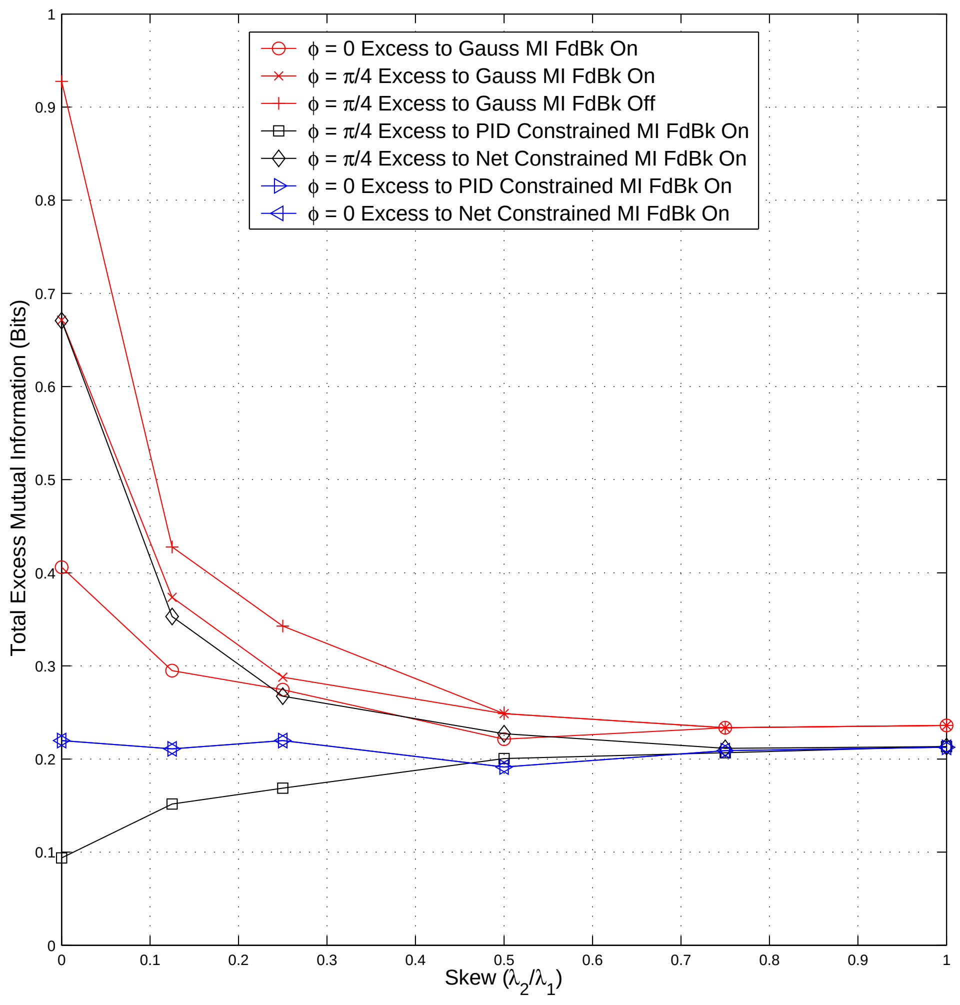


Figure 5.4. Excess mutual information at BER = 10^{-4} as measured against Gassian signaling, Net QPSK constrained capacity, and PID constrained QPSK capacity across eigenskew and two distinct values of ϕ.

that are sufficient to achieve a decoded BER rate of 10^{-4}. The worst case channel clearly occurs when $\phi = \pi/4$. For this channel, results with and without feedback between the decoder and detector show that using feedback yields more than a 0.2 bit improvement. The next two curves show the excess mutual information on the worst case channel ($\phi = \pi/4$) when mutual information is determined as PID constrained MI or Net constrained MI. Recall the PID constrained MI provides a lower bound on operational mutual information and Net constrained MI is an upper bound. Referring to Fig. 5.3 Net and PID MI are the same on the $\phi = \pi/4$ channel when $\kappa = 1$ and differ by about 0.325 bits when $\kappa = 0$. However, the mutual informations for the $\kappa = 0$ channel in Fig. 5.3 where computed using and SNR of 5.3 dB. The SNR had to be increased to 7.8 dB before the code would operate on this channel at a BER of 10^{-4}. At this SNR, the Net constrained mutual information provided by the $\phi = \pi/4$, $\kappa = 0$ channel is 2.0 bits, while the PID constrained MI is 1.42 bits. This explains the 0.58 bit decrease in performance when measuring against Net constrained versus PID constrained capacity. The two final curves on the plot measure the performance of the code on the $\phi = 0$ channel across eigenskew. The flatness of both the Net and the PID constrained curves is indicative of the congruence of these two mutual information measures at all eigenskews when $\phi = 0$ (Fig. 5.3).

We note that code performance very closely tracks available operating mutual information. Specifically, 'worst-case' channels yield performance that is not as good as 'best-case' channels because the operational PID constrained capacity on these channels is reduced. The last two curves in Fig. 5.4 show that when operational PID constrained capacity equals Net constrained capacity ($\phi = 0$) that total excess mutual information remains constant as κ varies between zero and one. Therefore, the code itself is robust to channel variation. The detection process, however, is the root cause of the performance degradation observed in the other cases.

5.3 Reducing the Complexity of Iterative Detection and LDPC Decoding

In this section we describe the basic principles of Turbo-BLAST, focusing on the different soft MIMO detectors. In Turbo-BLAST, an iterative detection and decoding receiver is used to approach the maximum-likelihood (ML) performance of joint MIMO detection and LDPC decoding. Fig. 5.5 gives a flowchart of the turbo iterative receiver structure. In this structure, the soft MIMO detector incorporates extrinsic information provided by the LDPC decoder, and the LDPC decoder incorporates soft information provided by the MIMO detector. Extrinsic information between the detector and decoder is then exchanged in an iterative fashion until an LDPC codeword is found or a maximum number of iterations is performed. With LDPC codes, convergence to a codeword is easy to detect since we need only verify that the parity checks are satisfied. The message-passing (also known as belief-propagation) decoding algorithm used to decode the LDPC code is described in detail in [38].

In the following we provide an overview of the soft MAP MIMO detector and the MMSE-SIC detector, and introduce two reduced-complexity detectors, an MMSE suppression detector and an MMSE hard decision interference cancellation detector. We

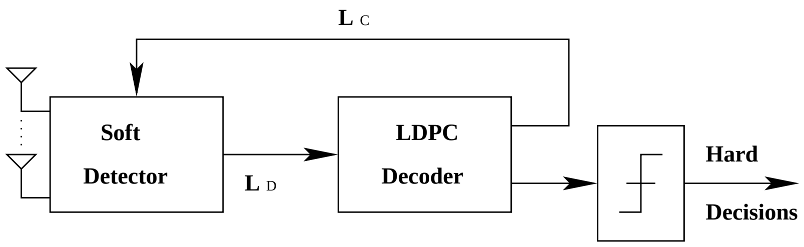


Figure 5.5. Turbo iterative detection and decoding receiver for an LDPC coded BLAST system.

examine the complexity of each scheme and select specific parameters for modulation cardinality and transmit-receive antenna multiplicity to facilitate a numerical comparison of complexity.

5.3.1 MAP Detector

In the soft MAP detector, the received vector $\mathbf{y}$ is de-mapped by a log-likelihood ratio (LLR) calculation for each bit included in the transmit vector $\mathbf{x}$ ($n_T M_c$ of them). The *extrinsic* information provided by the MAP detector is the difference of the soft-input and soft-output LLR values on the coded bits. For the ith code bit b_i ($i = 1, \dots, n_T M_c$) of the transmit vector $\mathbf{x}$, the extrinsic LLR value of the estimated bit is computed as

$$\begin{aligned} L_D(b_i) &= \log \frac{P(b_i = +1|\mathbf{y})}{P(b_i = -1|\mathbf{y})} - \log \frac{P(b_i = +1)}{P(b_i = -1)} \\ &= \log \frac{\sum_{\mathbf{x} \in \mathcal{X}_i^{+1}} P(\mathbf{y}|\mathbf{x}) P(\mathbf{x})}{\sum_{\mathbf{x} \in \mathcal{X}_i^{-1}} P(\mathbf{y}|\mathbf{x}) P(\mathbf{x})} - L_C(b_i) \end{aligned} \tag{5.8}$$

where $L_C(b_i)$ is the extrinsic information of the bit b_i computed by the LDPC decoder in the previous turbo iteration ($L_C(b_i) = 0$ at the first iteration) and $\mathcal{X}_i^{+1}$ is the set of

$2^{n_T Mc-1}$ vector hypotheses $\mathbf{x}$ having $b_i = +1$, ($\mathcal{X}_i^{-1}$ is similarly defined). Based on the extrinsic LLR of the code bits provided by the LDPC decoder and assuming the bits within $\mathbf{x}$ are statistically independent of one another, the *a priori* probability $P(\mathbf{x})$ can be written as

$$P(\mathbf{x}) = \prod_{j=1}^{n_T M_c} P(b_j) = \prod_{j=1}^{n_T M_c} \left[1 + \exp(-\mathbf{x}_{[j]} L_C(b_j))\right]^{-1} \tag{5.9}$$

where $\mathbf{x}_{[j]}$ corresponds to the value $(+1, -1)$ of the jth bit in the vector $\mathbf{x}$.

In the above LLR value calculation, the likelihood function $P(\mathbf{y}|\mathbf{x})$ is specified by the multi-dimensional Gaussian pdf

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{(2\pi\sigma^2)^{n_R}} \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2\right). \tag{5.10}$$

Note that for the LLR value calculation, only the term in the exponent is relevant; the constant factor outside the exponent can be dropped.

Since the cardinality of the vector sets $\mathcal{X}_i^{+1}$ and $\mathcal{X}_i^{-1}$ in (5.8) equals $2^{n_t M_c - 1}$, the complexity of the soft MAP detector is exponential in the number of transmitter antennas and the number of bits per constellation symbol. At each iteration, the MAP detector has to compute the LLRs for $n_T M_c$ bits in each transmit symbol vector. With an equal number of transmit and receive antennas, $n_T = n_R = n$, evaluating (5.8) involves the following steps:

```
for $i = 1$ to $2^{nM_c}$
    1) Compute the a priori probability $P(\mathbf{x})$ in (5.9): $2nM_c$ flops.
end
```

```
for $i = 1$ to $nM_c$
    2) Compute the LLR value in (5.8): $2 \cdot 2^{nM_c}$ flops.
end
```

In the above we do not consider the computation of the likelihood function $P(\mathbf{y}|\mathbf{x})$. This is precomputed for all 2^{nM_c} hypotheses at the beginning of the iterative process and has a cost of $2n^2 + 3n$ flops per vector hypothesis. We define a flop as a single addition, subtraction, multiplication or division between two complex numbers. Table lookups to $\exp$ and $\log$ functions are not included in this complexity analysis. Then, the (approximate) cost of the MAP detector per turbo iteration is $(4nM_c) \cdot 2^{nM_c}$ flops and the initial cost of precomputing the likelihood functions for all hypotheses is $(2n^2 + 3n) \cdot 2^{nM_c}$ flops.

5.3.2 MMSE-SIC Detector

The suboptimal demodulator based on a minimum-mean square-error soft-interference cancellation criterion is described in detail in [41] and [27]. Below we give a short review and comment on its complexity.

The MMSE-SIC detector first forms soft estimates of the symbols transmitted from the jth antenna ($j = 1, \ldots, n_T$) as

$$\tilde{x}_j = \sum_{x \in \mathcal{S}} x P(x) \qquad (5.11)$$

where $\mathcal{S}$ is the complex constellation set and

$$P(x) = \prod_{l=1}^{M_c} [1 + \exp(-x_{[l]} L_C(b_{(j-1)M_c + l}))]^{-1} \qquad (5.12)$$

where $x_{[l]}$ indicates the value of the lth bit of symbol x.

Then, for the kth antenna, the soft interference from the other $n_T - 1$ antennas is canceled to obtain

$$\mathbf{r}_k = \mathbf{y} - \sum_{j=1, j\neq k}^{n_T} \tilde{x}_j \mathbf{h}_j = x_k \mathbf{h}_k + \sum_{j=1, j\neq k}^{n_T} (x_j - \tilde{x}_j)\mathbf{h}_j + \mathbf{n}. \tag{5.13}$$

A detection estimate z_k of the transmitted symbol on the kth antenna is obtained by applying a linear MMSE filter $\mathbf{w}_k$ to $\mathbf{r}_k$,

$$z_k = \mathbf{w}_k^\dagger \mathbf{r}_k = (\mathbf{w}_k^\dagger \mathbf{h}_k) x_k + \sum_{j=1, j\neq k}^{n_T} (\mathbf{w}_j^\dagger \mathbf{h}_j)(x_j - \tilde{x}_j) + \mathbf{w}_k^\dagger \mathbf{n} \tag{5.14}$$

where † represents the conjugate transpose operator. The filter $\mathbf{w}_k$ is chosen to minimize the mean-square error between the transmit symbol x_k and the filter output z_k and is given by

$$\mathbf{w}_k = \left(\frac{n_T}{\rho} \mathbf{I}_{n_R} + \mathbf{H} \boldsymbol{\Delta}_k \mathbf{H}^\dagger \right)^{-1} \mathbf{h}_k \tag{5.15}$$

where the covariance matrix $\boldsymbol{\Delta}_\mathbf{k}$ is

$$\boldsymbol{\Delta}_k = \mathrm{diag}\{1 - |\tilde{x}_1|^2, \ldots, 1 - |\tilde{x}_{k-1}|^2, 1, 1 - |\tilde{x}_{k+1}|^2, \ldots, 1 - |\tilde{x}_{n_T}|^2\}. \tag{5.16}$$

Observe that in (5.14), the first term represents the desired term, the second term is residual interference from the other transmitter antennas and the last term is a phase rotated noise term. As noted in [49], we can make the following Gaussian approximation for the MMSE filter output z_k

$$z_k \sim \mathcal{N}(x_k, \eta_k^2) \tag{5.17}$$

where the variance η_k^2 is given by

$$\eta_k^2 = 1 - \mathbf{w}_k^\dagger \mathbf{h}_k. \tag{5.18}$$

Then, the extrinsic log-likelihood ratio computed by the MMSE-SIC detector for the lth bit ($l = 1, \ldots, M_c$) of the symbol x_k transmitted by the kth antenna ($k = 1, \ldots, n_T$) is

$$\begin{aligned} L_D(b_{(k-1)M_c+l}) &= \log \frac{P(b_{(k-1)M_c+l} = +1|z_k)}{P(b_{(k-1)M_c+l} = -1|z_k)} - \log \frac{P(b_{(k-1)M_c+l} = +1)}{P(b_{(k-1)M_c+l} = -1)} \\ &= \log \frac{\sum_{x \in \mathcal{S}_l^{+1}} P(z_k|x)P(x)}{\sum_{x \in \mathcal{S}_l^{-1}} P(z_k|x)P(x)} - L_C(b_{(k-1)M_c+l}) \end{aligned} \tag{5.19}$$

where $\mathcal{S}_l^{+1}$ is the set of 2^{M_c-1} hypotheses x for which the lth bit is $+1$ ($\mathcal{S}_l^{-1}$ is similarly defined). In the above calculation of the extrinsic LLR value, the *a priori* probability $P(x)$ is given by (5.12) and the likelihood function $P(z_k|x)$ is approximated by

$$P(z_k|x) \simeq \frac{1}{\pi\eta_k^2} \exp\left(-\frac{1}{\eta_k^2}\|z_k - x\|^2\right). \tag{5.20}$$

Note that the MMSE-SIC detector has a lower complexity than the MAP detector. This can be seen from (5.19) where the extrinsic LLR is computed from the *scalar* output z_k of the MMSE filter, in contrast with (5.8) where the extrinsic LLR is computed from the received vector $\mathbf{y}$. With an equal number of transmit and receive antennas, $n_T = n_R = n$, evaluating (5.19) involves the following steps:

```
for $k = 1$ to $n$
    for $i = 1$ to $2^{M_c}$
        1) Compute the a priori probability $P(x)$ in (5.12): $2M_c$ flops.
    end
    2) Evaluate the soft estimate in (5.11): $2 \cdot 2^{M_c}$ flops.
    3) Cancel the soft estimates in (5.13): $2n^2$ flops.
    4) Evaluate the matrix $\mathbf{G} = \left(\frac{n_T}{\rho}\mathbf{I}_{n_R} + \mathbf{H}\boldsymbol{\Delta}_k\mathbf{H}^\dagger\right)$: $2n^3 + n^2 + 3n$ flops.
    5) Solve $\mathbf{G}\mathbf{w}_k = \mathbf{h}_k$ for $\mathbf{w}_k$: $n^3/3$ flops using Choleski factorization [17].
    6) Compute the detection estimate $z_k$: $2n$ flops.
    for $l = 1$ to $M_c$
        7) Compute the LLR value in (5.19): $2 \cdot 2^{M_c}$ flops.
    end
end
```

Then, the (approximate) computational complexity of the MMSE-SIC detector is $7n^4/3 + 3n^3 + 5n^2 + (4nM_c) \cdot 2^{M_c} + (2n) \cdot 2^{M_c}$ flops per turbo iteration.

5.3.3 MMSE Suppression Detector

While the MMSE-SIC detector reduces the complexity of computing the extrinsic LLR values from exponential to polynomial in the number of transmit antennas, it still has the undesirable cost of re-evaluating the MMSE filter $\mathbf{w}_k$ and updating z_k with each turbo iteration, as these depend on the soft estimates from the LDPC decoder in the previous iteration. A simple linear MMSE suppression filter can significantly reduce the complexity of the soft MIMO detector and incurs only a small performance degradation as compared to MMSE-SIC.

In this case the MMSE suppression filter is evaluated only once, at the beginning of the turbo iterative process, and applied to the received vector $\mathbf{y}$ to obtain a detection estimate for the kth antenna,

$$z_k = \mathbf{w}_k^{\dagger}\mathbf{y}. \tag{5.21}$$

No interference cancellation is performed and the linear MMSE filter $\mathbf{w}_k$ is chosen to suppress the co-antenna interference,

$$\mathbf{w}_k = \left(\frac{n_T}{\rho} \mathbf{I}_{n_R} + \mathbf{H}\mathbf{H}^{\dagger} \right)^{-1} \mathbf{h}_k. \tag{5.22}$$

As before, we make the Gaussian approximation of the soft MMSE filter output z_k, with the variance η_k^2 as in equation (5.18), where $\mathbf{w}_k$ is replaced by the new MMSE filter. The extrinsic LLR values are computed exactly as in the case of the MMSE-SIC detector using (5.19). Assuming $n_T = n_R = n$, evaluating the LLR values involves the following steps:

```
for $k = 1$ to $n$
    for $i = 1$ to $2^{M_c}$
        1) Compute the a priori probability $P(x)$ in (5.12): $2M_c$ flops.
    end
    for $l = 1$ to $M_c$
        2) Compute the LLR value in (5.19): $2 \cdot 2^{M_c}$ flops.
    end
end
```

Then, the complexity per turbo iteration is $(4nM_c) \cdot 2^{M_c}$ flops. Moreover, the initial cost of evaluating the MMSE suppression filters $\mathbf{w}_k$ and the detection estimates z_k is $7n^4/3 + 3n^2$ flops. Note that the MMSE suppression detector reduces the complexity per turbo iteration from $\mathcal{O}(n^4)$ (for the MMSE-SIC detector) to $\mathcal{O}(n)$. Of course, the overall complexity for the MMSE suppression scheme remains $\mathcal{O}(n^4)$ because of required initial processing. However, a more flexible allocation of computational resources becomes possible given the need to solve a system of equations to determine $\mathbf{w}_k$ only once rather than on a per iteration basis.

5.3.4 MMSE-HIC Detector

Here we introduce another reduced complexity detector based on the simple MMSE filter with hard decision interference cancellations. The main idea is to improve the performance of the MMSE suppression detector by canceling hard decision estimates of the interfering symbols while also maintaining the reduced computational complexity. At the beginning of the iterative process, cancellations are not possible since no reliability information is yet available from the decoder. Therefore, in the first iteration, we use the MMSE suppression filter $\mathbf{w}_k$ from (5.22) in the same manner as above to obtain the detection estimate z_k for the kth antenna. However, in subsequent iterations, as soft information from the decoder becomes available, hard decision estimates on the LDPC code bits b_i ($i = 1, \ldots, n_T M_c$) can be obtained from

$$\hat{b}_i = \text{sign}\left[L_D(b_i) + L_C(b_i)\right] \tag{5.23}$$

and a hard decision estimate on the kth antenna symbol can be formed as

$$\hat{x}_k = f\left(\hat{b}_{(k-1)M_c+1}\hat{b}_{(k-1)M_c+2}\ldots\hat{b}_{kM_c}\right) \tag{5.24}$$

where f is a function that maps an input bit vector to a complex constellation point. Assuming these hard decision symbol estimates are correct, their cancellation would provide a better detection estimate for the antenna of interest. More specifically, for the kth transmit antenna, the hard decision estimates of the other $n_T - 1$ interfering antennas can be canceled to obtain

$$\mathbf{r}_k = \mathbf{y} - \sum_{j=1, j\neq k}^{n_T} \hat{x}_j \mathbf{h}_j \tag{5.25}$$

and the new detection estimate z_k can be obtained by maximum-ratio-combining,

$$z_k = \frac{\mathbf{h}_k^\dagger \mathbf{r}_k}{\|\mathbf{h}_k\|^2 + \frac{n_T}{\rho}}. \tag{5.26}$$

Observe that the assumption of correct hard decisions does not hold very well especially in the early stages of the iterative process. In order to avoid error propagation due to incorrect hard decisions, it is very important that cancellations be performed only when the reliability of the canceled symbols is high according to some cancellation criterion. We experimented with different criteria for interference cancellation and found that the following two methods give the best results:

- **Average of LLRs**

  With this criterion, first the following average is computed and then compared to a predetermined threshold value

  $$\Phi_a = \frac{1}{n_T M_c} \sum_{i=1}^{n_T M_c} |L_C(b_i)| \geq \Theta_a. \tag{5.27}$$

  The threshold Θ_a is found experimentally as the threshold that yields the best bit-error rate (BER) performance. Observe that a too low threshold value would introduce undesirable error propagation due to incorrect cancellations, while a too high threshold value would give the same performance as the MMSE suppression detector since no cancellations are performed in this case.

- **Probability of bit vector**

  With this criterion, the probability of a bit vector is first computed and then compared to a threshold value

$$\Phi_p = P(b_1 b_2 \dots b_{n_T M_c}) = \prod_{i=1}^{n_T M_c} \left[1 + \exp(-|L_C(b_i)|)\right]^{-1} \geq \Theta_p. \tag{5.28}$$

  As before, the threshold Θ_p is found experimentally to optimize the BER performance.

The computational complexity of the MMSE-HIC detector is that of the MMSE suppression detector plus the additional cost of checking the cancellation criterion every turbo iteration and performing the hard decision interference cancellation whenever the criterion is satisfied. Assuming $n_T = n_R = n$, evaluating the LLR values involves the following steps:

```
for $k = 1$ to $n$
    1) Check the cancellation criterion in (5.27): $nM_c$ flops.
    2) Cancel the hard decision estimates in (5.25): $2n^2$ flops.
    3) Compute the detection estimate $z_k$ in (5.26): $4n$ flops.
    for $i = 1$ to $2^{M_c}$
        4) Compute the a priori probability $P(x)$ in (5.12): $2M_c$ flops.
    end
    for $l = 1$ to $M_c$
        5) Compute the LLR value in (5.19): $2 \cdot 2^{M_c}$ flops.
    end
    end
```

Then, the worst-case computational complexity (assuming cancellations are performed every iteration) is $(2n^3 + n^2 M_c + 4n^2) + (4nM_c) \cdot 2^{M_c}$ flops per turbo iteration. The first terms in this summation represent the additional complexity over the MMSE suppression detector. As in that case, there is an initial cost of $7n^4/3 + 3n^2$ flops to evaluate the MMSE filters and the detection estimates.

In Table 5.2 we give a complexity comparison example based on a flop count for the MAP, MMSE-SIC, MMSE suppression and MMSE-HIC detectors for $n \times n$ antenna configurations with $n = 2, 4, 8$, using QPSK modulation. For each detector, we provide the initial cost (which is the cost of computations that must be done only once for all of the iterations), the cost per turbo iteration, and the total computational cost assuming 30 iterations.

5.4 Performance Comparison of the Different Detectors

5.4.1 MI Performance of Different Detectors on parameterized 2×2 Channels

Points plotted as circles in Fig. 5.6 illustrate the 2×2 total excess MI (assuming Gaussian alphabet signaling) for the $\mathbf{a} = [1, 1]$ optimized code of Table 5.1. Again, per Fig. 5.3, operational mutual information varies more significantly with parameter ϕ as κ approaches zero. This variation is manifested as a performance loss on $\phi = \pi/4$ channels and is most severe for the MMSE-Supression detector. This is the simplest of

	$n = 2$	$n = 4$	$n = 8$
MAP initial	224	1.13×10^4	9.96×10^6
MAP per iteration	256	8.19×10^3	4.19×10^6
MAP total (30 iterations)	7.90×10^3	2.57×10^5	1.36×10^8
MMSE-SIC initial	0	0	0
MMSE-SIC per iteration	161	1.03×10^3	1.17×10^4
MMSE-SIC total (30 iterations)	4.83×10^3	3.09×10^4	3.51×10^5
MMSE suppression initial	49	645	9.75×10^3
MMSE suppression per iteration	64	128	256
MMSE suppression total (30 iterations)	1.97×10^3	4.48×10^3	1.74×10^4
MMSE-HIC initial	49	645	9.75×10^3
MMSE-HIC per iteration	104	352	1.66×10^3
MMSE-HIC total (30 iterations)	3.17×10^3	1.12×10^4	5.97×10^4

Table 5.2. Cost (in flops) of computing the LLRs for different MIMO detectors.

the detectors as it does not employ feedback between the decoder and the detector. Note that the MAP detector with feedback turned off (Fig. 5.4) exhibited a similar, though less severe, performance loss on the $\kappa = 0$, $\phi = \pi/4$ channel.

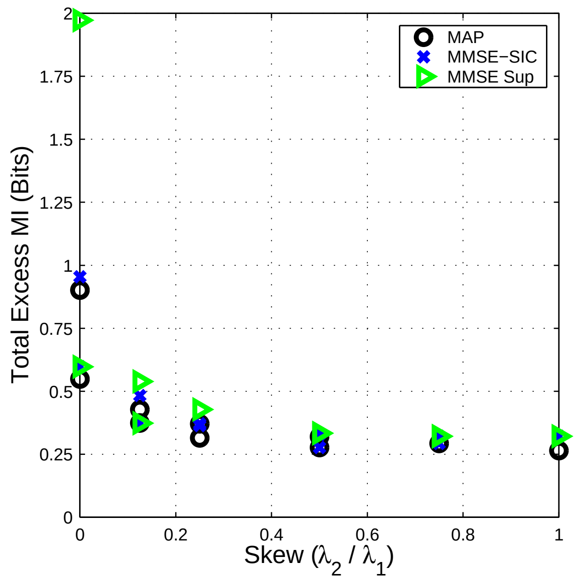


Figure 5.6. Simulation results showing excess MI vs. eigenvalue skew κ at BER $= 10^{-4}$ for different detectors (rate 1/3 length 15,000 a $=$ $[1, 1]$ optimized code modulating QPSK on a 2×2 channel). The MAP and MMSE-SIC detectors perform similarly with worst case channels occurring when $\kappa = 0$ under the $\phi = \pi/4$ rotation. The MMSE only detector suffers the most severe degradation on the $\kappa = 0$, $\phi = \pi/4$ since feedback is not employed to suppress the co-channel interference present under this parameterization.

5.4.2 SNR and MI Performance of Different Detectors in Fast Rayleigh Fading

In this section we examine the performance of LDPC coded BLAST systems using the soft MIMO detectors introduced in the previous section. In our study, we assume that the number of receive antennas is the same as the number of transmit antennas, i.e. $n_R = n_T$. The LDPC code used in the simulations is the rate-$1/3$, length 15000 code that was realized from the degree distribution given by the code labeled $\mathbf{a} = [1, 1]$ in Table 5.1. The density evolution threshold for this code in AWGN is -4.92 dB. This degree sequence was found via a linear program that sought the highest rate ensemble under a given threshold and maximum left and right node degree constraints [11]. In order to acquire a given rate goal, the density evolution initial mean was adjusted to achieve the desired rate. The mapping in all our simulations is a Gray-labeled QPSK constellation. The resulting spectral efficiency is $4/3$ bits/channel use.

We assume a fast Rayleigh fading scenario, where the channel matrix is realized independently from one transmission time to the next. We compare our bit-error rate results with the theoretical channel capacity limit. Under the fast fading assumption, the theoretical capacity limit is the ergodic channel capacity given by [42],

$$C = E\left[\log_2 \det\left(\mathbf{I}_{n_R} + \frac{\rho}{n_T}\mathbf{H}\mathbf{H}^\dagger\right)\right] \tag{5.29}$$

where the expectation is over the entries of $\mathbf{H}$.

Figs. 5.7, 5.8, 5.9, and 5.10 show the BER performance versus average SNR per receiver antenna on 2×2, 3×3, 4×4, and 8×8 fast fading Rayleigh channels. On these plots we also show the channel capacity at the corresponding transmission bit rate for

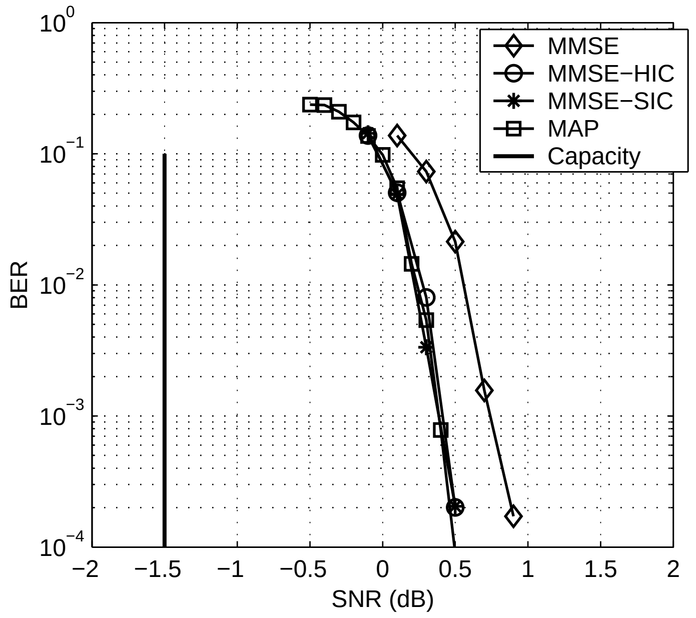


Figure 5.7. Performance of LDPC coded BLAST for 2×2 MIMO system with MAP, MMSE-SIC, MMSE-HIC and MMSE suppression detectors.

these systems. As expected, the MAP detector yields the best performance, which is 2 dB away from the theoretical capacity at BER $= 10^{-4}$ on the 2×2 channel. We note that the MMSE-SIC detector has essentially the same performance as the MAP detector on the 2×2, 3×3, and 4×4 systems. For the 8×8 channel the computational complexity associated with the MAP detector is prohibitive and this result was not simulated.

Our results differ from the results reported in [27] where a system based on the MMSE-SIC detector has a performance degradation (less than 1 dB) compared to a system based on the MAP detector. We attribute this difference to the fact that in [27] a detector iteration is performed only after a number of decoder iterations, whereas in our work every decoder iteration is followed by a detector iteration.

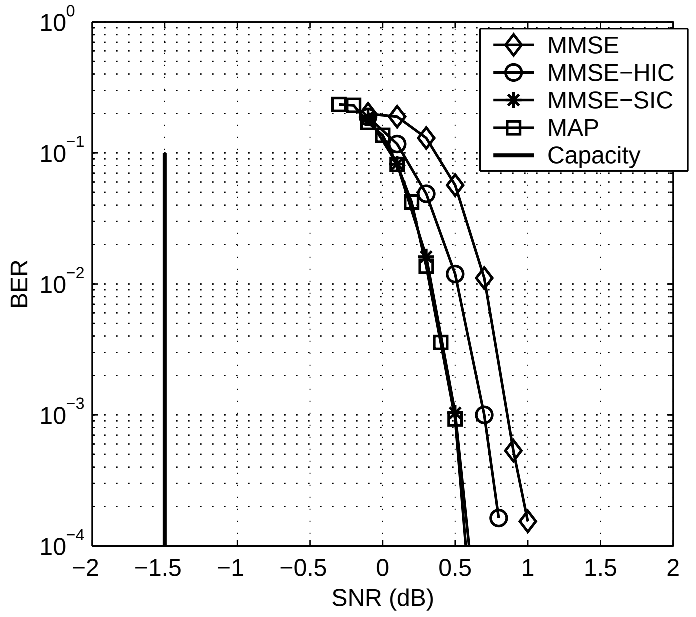


Figure 5.8. Performance of LDPC coded BLAST for 3×3 MIMO system with MAP, MMSE-SIC, MMSE-HIC and MMSE suppression detectors.

Moreover, our simulation results show that the very low complexity MMSE suppression detector has a performance loss of only 0.5 dB or less. This loss is approximately cut in half if the MMSE-HIC detector with an optimized cancellation threshold is used instead. In fact, in the 2×2 system, the MMSE-HIC detector performance is very close to the MAP and MMSE-SIC detectors.

Though the majority of this section has focused on a comparison of the performance of the different detectors in terms of SNR in Rayleigh fast fading we again emphasize the robustness of the codes as measured in terms of excess mutual information via Fig. 5.11.

The cases plotted in this figure are contrasted from the exhaustively parameterized

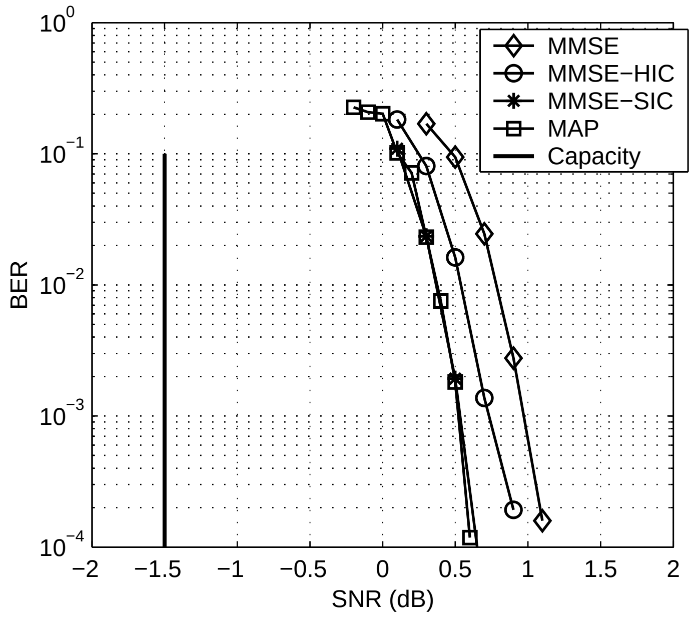


Figure 5.9. Performance of LDPC coded BLAST for 4×4 MIMO system with MAP, MMSE-SIC, MMSE-HIC and MMSE suppression detectors.

2×2 case where we were able to determine best and worst case channels. Here, instead, all channel results are average together in conjunction with the Rayleigh distribution. Note that under MAP and MMSE-SIC detection, the excess MI per antenna remains essentially constant, at 0.25 bits, as the number of transmit/receive antennas increases from 2 to 4. MAP detector results were not computed (due to prohibitive complexity) for the 8×8 case.

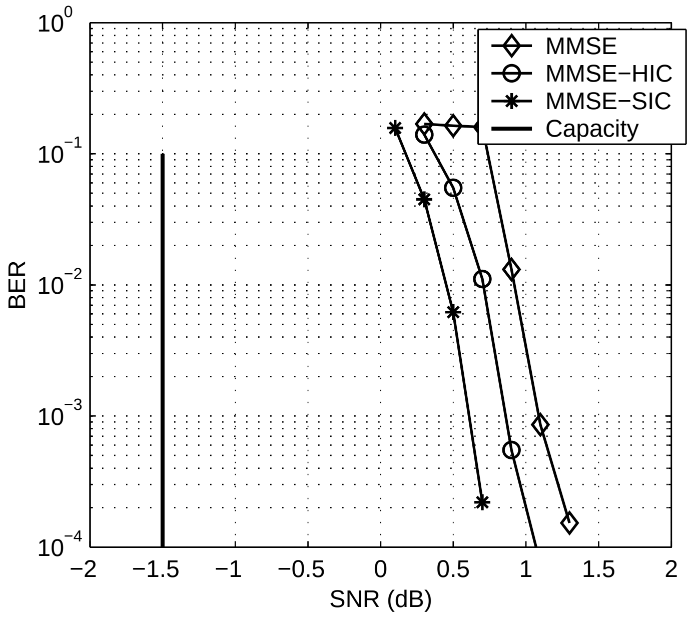


Figure 5.10. Performance of LDPC coded BLAST for 8×8 MIMO system with MMSE-SIC, MMSE-HIC and MMSE suppression detectors.

5.5 Conclusion

In this chapter we have shown that the excess mutual information required to achieve a bit error rate of 10^{-4} on the 2×2 channel with a particular LDPC code varies by approximately 0.5 bits (with a total rate 4/3 code) between the best and worst case channel when excess MI is measured against Net constrained capacity. However, excess MI is essentially constant when measured against PID constrained capacity. Code performance very closely tracks available operating mutual information. Therefore, LDPC codes themselves are robust to channel variation. The information loss incurred during joint PDF marginalization to individual bit PDFs in the detection process is the root cause of excess mutual information variation with eigenskew.

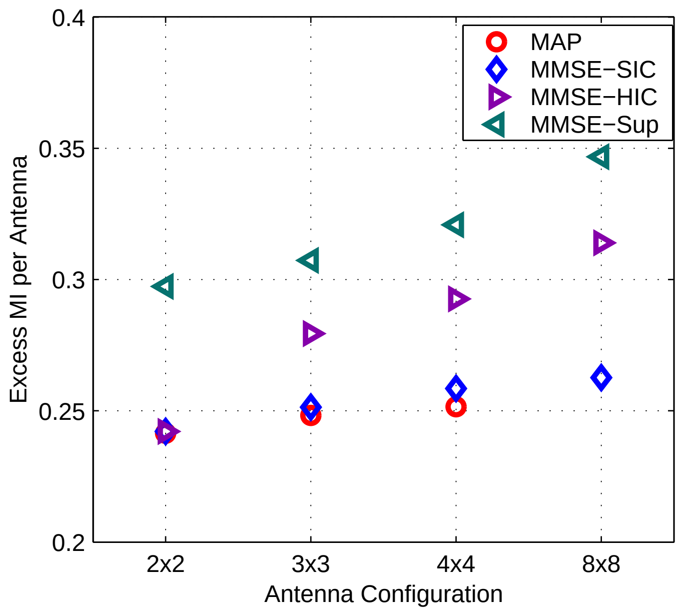


Figure 5.11. Performance of different detectors across increasing antenna multiplicities in terms of excess mutual information per transmit antenna in Rayleigh fading. Each excess MI is measured from constrained ergodic Rayleigh capacity for the given channel.

We have also shown that several variants of LDPC coded BLAST systems with iterative detection and decoding perform roughly 2 dB from the theoretical capacity of the Rayleigh MIMO channel. The soft MAP detector is optimal, however the MMSE-SIC detector exhibits similar performance and has a lower overall computational complexity. The MMSE suppression and MMSE-HIC detectors offer yet lower complexities, for relatively small performance penalties. As an example, for the 4×4 system, considering the MAP detector as the norm, the total complexity (with 30 iterations) of the MMSE-SIC detector is 8 times lower, that of the MMSE-HIC detector is 23 times lower, and that of the MMSE suppression detector is 57 times lower. A performance loss of around 0.5 dB is observed for the MMSE suppression detector. This loss is cut approximately in half by an MMSE-HIC detector using a carefully chosen cancellation threshold.

Appendix - Operational Mutual Information in Systems Without Prior Information Feedback

Theorem 3 *The mutual information available to an iterative decoder for the case of a generalized Gaussian channel without feedback between the detector and decoder is given by*

$$\tilde{I}(\mathbf{X};\mathbf{Y}|\mathbf{H}) = \sum_{j=0}^{N_t M_c - 1} I(\mathbf{X}(b_j);\mathbf{Y}|\mathbf{H}), \tag{5.30}$$

where $\mathbf{X}(b_j)$ calls out the jth bit in the transmit vector $\mathbf{X}$ which carries a total of $N_t M_c$ bits, and $\mathbf{X}$,$\mathbf{Y}$ are RVs and $\mathbf{H}$ is the given channel matrix.

Proof: Consider the decoding scenario where the *instantaneous* mutual information entering the decoder due to each received vector $\mathbf{y}_k$ is given by $\hat{I}_k$ (here we re-introduce time index k),

$$\begin{aligned} \hat{I}_k &= \sum_{j=0}^{N_t M_c - 1} \hat{I}^{b_j}, \quad \text{where} \\ \hat{I}^{b_j} &= h\left(p\left(\mathbf{x}_k(b_j)\right)\right) - h\left(p\left(\mathbf{x}_k(b_j)|\mathbf{y}_k,\mathbf{H}_k\right)\right). \end{aligned} \tag{5.31}$$

Observe that $\hat{I}_k$ is the sum of the instantaneous marginalized bit reliability mutual informations. The marginalization step reduces the information entering the decoder at each receive time k from the instantaneous vector-wise joint mutual information, $I\left(\mathbf{x}_k; \mathbf{y}_k | \mathbf{H}_k\right)$, to the sum of the instantaneous bitwise mutual informations, or $\hat{I}_k$. Explicitly, $\hat{I}_k$ is defined as (for uniform $p(\mathbf{x}_k(b_j))$),

$$\hat{I}_k = \sum_{j=0}^{N_t M_c - 1} \begin{pmatrix} \sum_{b_j=\{0,1\}} p\left(\mathbf{x}_k(b_j) | \mathbf{y}_k, \mathbf{H}_k\right) \\ \log_2\left(p\left(\mathbf{x}_k(b_j) | \mathbf{y}_k, \mathbf{H}_k\right)\right) \\ -\log_2(1/2) \end{pmatrix} \tag{5.32}$$

To complete the proof, observe that the expectation of the instantaneous mutual information converges to the ergodic mutual information of the sum of the individual bit planes in the limit of infinite sample size n,

$$\frac{1}{n}\sum_{k=0}^{n-1} \hat{I}_k \xrightarrow[n\to\infty]{} \sum_{j=0}^{N_t M_c - 1} I\left(\mathbf{X}(b_j); \mathbf{Y} | \mathbf{H}\right). \tag{5.33}$$

An exchange in the order of the two outermost summations in (5.33) left-hand-side and (5.32) respectively yields this asymptotic result. Hence the average given by (5.30) is the information available to the *closed* iterative decoder.

■

Bibliography

[1] A. A. AlRustamani, A. Stefanov, and B. R. Vojcic. Turbo-greedy coding for multiple antenna systems. In *Proceedings of ICC*, volume 6, pages 1684 –1689, June 2001.

[2] D. M. Arnold, E. Eleftheriou, and X. Y. Hu. Progressive edge-growth Tanner graphs. in *Proc. IEEE Global Telecommun. Conf.*, San Antonio, TX, Nov. 2001, 2:995–1001.

[3] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara. Serial concatenation of interleaved codes: performance analysis, design, and iterative decoding. *IEEE Trans. Inform. Theory*, 44:909–926, May 1998.

[4] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: Turbo codes. in *Proc. IEEE Int. Conf. Commun.*, Geneva, Switzerland, pages 1064–1070, May 1993.

[5] Jones C., Matache A., Tian T., Villasenor J., and Wesel R. D. The universality of ldpc codes on wireless channels. In *Proceedings of MilCom*, Oct 2003.

[6] Jones C., Tian T., Matache A., Wesel R. D., and Villasenor J. Robustness of ldpc codes on periodic fading channels. In *Proceedings of GlobeCom*, Nov 2002.

[7] J. Chen, A. Dholakia, E.Eleftheriou, M. Fossorier, and X.-Y. Hu. Near optimum reduced-complexity decoding algorithms for LDPC codes. in *Proc. IEEE Int. Sym. Inform. Theory,* Lausanne, Switzerland, Jul. 2002.

[8] J. Chen and M. Fossorier. Density evolution for BP-based decoding algorithms of LDPC codes and their quantized versions. in *Proc. IEEE Globecom,* Taipei, Taiwan, Nov. 2002.

[9] J. Chen and M. Fossorier. Near optimum universal belief propagation based decoding of LDPC codes. *IEEE Trans. on Comm.,* 50(3), Mar. 2002.

[10] A. Chini. *Multi Carrier Modulation in Frequency Selective Channels*. PhD thesis, Carleton Institute for Electrical Engineering, Ottawa, Sept. 1994.

[11] S.Y. Chung, T. Richardson, and R. Urbanke. Analysis of sum-product decoding of low-density parity-check codes using a gaussian approximation. *IEEE Trans. on Inform. Theory*, 47:657–670, Feb 2001.

[12] C. Di, D. Proietti, E. Telatar, T. Richardson, and R. Urbanke. Finite length analysis of low-density parity-check codes on the binary erasure channel. *IEEE Trans. Inform. Theory,* 48:1570–1579, June 2002.

[13] L. Dong-U, J. Villasenor, and W. Luk. A hardware Gaussian noise generator for

exploring channel code behavior at very low bit error rates. In *In proceedings of FCCM*, May 2003.

[14] E.Eleftheriou, T. Mittelholzer, and A. Dholakia. A reduced-complexity decoding algorithm for low-density parity-check codes. *IEE Electron. Letters*, 37:102–104, Jan. 2001.

[15] G. J. Foschini and M. J. Gans. On limits of wireless communications in a fading environment when using multiple antennas. *Wireless Personal Communications*, 6:311–335, 1998.

[16] C. Fragouli and R. D. Wesel. Bit vs. symbol interleaving for parallel concatenated trellis coded modulation. in *Proc. IEEE Global Telecommun. Conf.,* San Antonio, TX, Nov. 2001, 2:931–935.

[17] G. H. Golub and C. F. van Loan. *Matrix Computations*. Baltimore, MD: Johns Hopkins Univ. Press, third edition, 1996.

[18] R.G. Gallager. Low-density parity-check codes. *IRE Trans. Inform. Theory*, IT-8:21–28, Jan 1962.

[19] C. Jones, R. Tian, R. Wesel, and J. Villasenor. The Universal Operation of LDPC Codes in Scalar Fading Channels. *In Submission to the IEEE Transactions on Communications*, Oct 2003.

[20] J.H. Kang and W.E. Stark. Turbo codes for coherent fh-ss with partial band interference. In *Proceedings of Military Communications Conference*, Nov 1997.

[21] ”Köse, C., and R.” Wesel. Universal space-time trellis codes. In *Proceedings of GlobeCom*, Nov 2002.

[22] ”Köse, C., and R.” Wesel. Universal space-time trellis codes. *IEEE Transactions on Information Theory, Special Issue on Space-Time Transmission, Reception, Coding and Signal Design*, Nov. 2003.

[23] C. Kose, W.Y. Weng, and R.D. Wesel. Serially concatenated trellis coded modulation for the compound periodic erasures channel. In *Proceedings of ICC*, May 2003.

[24] Y. Kou, S. Lin, and M. Fossorier. Low-density parity-check codes based on finite geometries: a rediscovery and new results. *IEEE Trans. Inform. Theory*, 47:2711–2736, Nov. 2001.

[25] F.R. Kschischang, B.J. Frey, and H.A. Loeliger. Factor graphs and the sum-prodcut algorithm. *IEEE Trans. on Inform. Theory*, 47:498–519, Feb 2001.

[26] L. Lampe, R. Schober, and R. Fischer. Multilevel coding for multiple-antenna transmission. In *Proceedings of 2002 International Symposium on Information Theory*, volume 1, July 2002.

[27] B. Lu, G. Yue, and X. Wang. Performance analysis and design optimization of LDPC coded MIMO OFDM systems. *to appear in EURASIP JASP*, 51(11).

[28] M.G. Luby, M. Mitzenmacher, M.A. Shokrollahi, and D.A. Spielman. Improved

low-density parity-check codes using irregular graphs. *IEEE Trans. on Inform. Theory*, 47:569–584, Feb 2001.

[29] D.J.C. Mackay. Good error correcting codes bases on very sparse matrices. *IEEE Trans. on Inform. Theory*, 45:399–431, Mar 1999.

[30] D.J.C. MacKay, S.T. Wilson, and Davey M.C. Comparison of constructions of irregular Gallager codes. *IEEE Trans. on Comm.*, 47(10):498–519, October 1999.

[31] Y. Mao and A. H. Banihashemi. A heuristic search for good low-density parity-check codes at short block lengths. in *Proc. IEEE Int. Conf. Commun.,* Helsinki, Finland, June 2001.

[32] G. A. Margulis. Explicit construction of graphs without short cycles and low density codes. *Combinatorica*, 2(1):71–78, 1982.

[33] A. Matache and R. D. Wesel. Universal Trellis Codes for Diagonally Layered Space-Time Systems. *IEEE Transactions on Signal Processing, Special Issue on MIMO Wireless*, Nov. 2003.

[34] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Francisco, CA, 1998.

[35] M. Pursely and J. Skinner. Decoding strategies for turbo product codes in frequency-hop wireless communications. In *Proceedings of ICC*, May 2003.

[36] T. Richardson, A. Shokrollahi, and R. Urbanke. Design of capacity approaching

irregular low density parity check codes. *IEEE Trans. on Inform. Theory*, 47:618–637, Feb 2001.

[37] T. Richardson and R. Urbanke. Efficient encoding of low-density parity-check codes. *IEEE Trans. on Inform. Theory*, 47:638–656, Feb 2001.

[38] T. Richardson and R. Urbanke. The capacity of low-density parity-check codes under message passing decoding. *IEEE Trans. on Inform. Theory*, 47:599–618, Feb 2001.

[39] W.L. Root and P.P Varaiya. Capacity of classes of Gaussian channels. *SIAM J. Appl. Math*, 16(6):1350–1393, November 1968.

[40] M. Sellathurai and S. Haykin. TURBO-BLAST for high speed wireless communications. In *Proceedings of WCNC*, volume 1, pages 315–320, Sept 2000.

[41] M. Sellathurai and S. Haykin. Turbo-BLAST for wireless communications: theory and experiments. *IEEE Trans. Signal Proc.*, 50(10):2538–2546, Oct 2002.

[42] I. E. Telatar. Capacity of multi-antenna Gaussian channels. *European Trans. Telecommun.*, 10(6):585–595, Nov.-Dec. 1999. Available: `http://mars.bell-labs.com/cm/ms/what/mars/papers/proof/`.

[43] S. Ten Brink and B. Hochwald. Achieving near-capacity on a multiple-antenna channel. *IEEE Trans. on Comm.*, 51(3):389–399, Mar. 2003.

[44] Tian T., Jones C., Villasenor J., Wesel R. D. Characterization and selective avoid-

ance of cycles in irregular LDPC code construction. Submitted to *IEEE Trans. on Comm.*, 2002.

[45] Tian T., Jones C., Villasenor J., Wesel R. D. Characterization and selective avoidance of cycles in irregular ldpc codes. In *Proceedings of ICC*, May 2003.

[46] A. van Zelst, R. van Nee, and G. A. Awater. Turbo-BLAST and its performance. In *Proceedings of VTC*, volume 2, pages 1282–1286, May 2001.

[47] A. Vardy. The intractability of computing the minimum distance of a code. *IEEE Trans. Inform. Theory*, 43:1757–1766, Nov. 1997.

[48] U. Wachsmann, R. Fischer, and J. Huber. Multi-Level Codes: Theoretical Concepts and Practical Design Rules. *IEEE Trans. on Inform. Theory*, 45:1361–1391, May 1999.

[49] X. Wang and H. V. Poor. Iterative (Turbo) soft interference cancellation and decoding for coded CDMA . *IEEE Trans. on Comm.*, 47:1046–1061, July 1999.

[50] W.Y. Weng, C. Kose, and R.D. Wesel. Serially concatenated trellis coded modulation for the compound periodic erasures channel - in submission. *IEEE Trans. on Comm.*

[51] R. Wesel. *Trellis Code Design for Correlated Fading and Achievable Rates for Tomlinson-Harashima Precoding*. PhD thesis, Stanford University, Department of Electrical Engineering, Palo Alto, Aug. 1996.

[52] Wesel, R.D. and Cioffi, J.M. Trellis codes design for periodic interleavers. *IEEE Comm. Letters*, 3:103–105, April 1999.

[53] Wesel, R.D. and Liu, X. and Shi, W. Trellis codes for periodic erasures. *IEEE Trans. on Comm.*, 48:938–947, June 2000.

[54] M. Yang and W. E. Ryan. Lowering the error-rate floors of moderate-length high-rate irregular LDPC codes. In *In proceedings of IEEE Int. Symp. Infor. Theory,* Yokohama, Japan, June 2003.